

# Video on Demand Player SDK Product Documentation





#### Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

#### 🔗 Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

#### Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

### Contents

Player SDK

Overview

Basic Concepts

Features

Free Demo

Free Trial License

Purchase Guide

**Billing Overview** 

**Refund Policies** 

Overdue and Suspension Policy

SDK Download

SDK Download

Release Notes(Web)

Release Notes(iOS & Android)

Release Notes(Flutter)

Licenses

Adding and Renewing a License

Configuring and Viewing a License

Player Guide

Stage 1. Play back a source video

Stage 2. Play back a transcoded video

Stage 3. Play back an adaptive bitrate streaming video

Stage 4. Play back an encrypted video

Stage 5. Play back a long video

Integration (UI Included)

Web Integration

**TCPlayer Integration Guide** 

**TCPlayer Resolution Configuration Guide** 

TCPlayer Swift Live Streaming Downgrade Notice

iOS Integration Guide

Android Integration Guide

Flutter Integration Guide

Integration (No UI)

Web Integration

**TCPlayer Integration Guide** 

S Tencent Cloud

**TCPlayer Resolution Configuration Guide** TCPlayer Swift Live Streaming Downgrade Notice iOS Integration Integration Guide **VOD** Scenario Android Integration Integration Guide **VOD** Scenario Flutter Integration Integration Guide **VOD** Scenario **Advanced Features** Web Advanced Features Security Check Plugin (TCPlayerSafeCheckPlugin) VR Playback Plugin (TCPlayerVRPlugin) **Advanced Mobile Features** Picture-in-Picture Component (TUIPIP) iOS TUIPlayerShortVideo iOS Android **API** Documentation Web iOS Android Flutter **Player Adapter** Player Adapter for iOS Player Adapter for Android Player Adapter for Web **Player SDK Policy Privacy Policy** 

# Player SDK Overview

Last updated : 2023-08-16 11:26:55

### **Product Description**

Player SDK is a comprehensive, stable, and smooth video playback service provided for VOD businesses. It helps you connect to Tencent Cloud services and build cloud-based integrated capabilities. The Player and Player Adapter are offered for VOD playback scenarios and support various platforms including web, iOS, Android, and Flutter. In addition, it comes with a variety of video playback solutions to meet your diverse requirements in different business scenarios.

### **Quick Introduction**

To help you quickly understand the Player SDK, we recommend you first read Concepts of Player and quickly select the player type suitable for your business as instructed in How to Select Player before using Player and Player Adapter.

### Core Strengths

### **Cloud-based integration**

Player integrates powerful VOD audio/video service capabilities and provides comprehensive cloud-based integration features to offer your business more capabilities.

### All-around video security

Player supports video security solutions such as hotlink protection, URL authentication, HLS encryption, private protocol encryption, and offline download. It also features video security capabilities such as dynamic watermarking to



help protect the security of your media assets in different scenarios.

#### **Comprehensive data support**

Player supports monitoring the video playback quality over the entire linkage through metrics such as playback performance, user behaviors, and file characteristics.

### **Ultimate playback experience**

Player relies on Tencent Cloud's high numbers of cache nodes to provide complete video delivery acceleration capabilities with millisecond-level latency, delivering an ultrafast video playback experience.

### **Diverse playback capabilities**

Player provides various features such as instant broadcasting of the first frame, buffering during playback, adjustablespeed playback, video timestamping, on-screen commenting, and addon subtitles.

### **Use Cases**

### Short video playback

By integrating various VOD features, including content moderation, media asset management, seamless switch, instant broadcasting of the first frame, and interactive floating window, Player is often used in UGSV application development. For more information, see Demo.

### Long video playback

Player integrates VOD features such as adaptive bitrate streaming, seamless definition switch, thumbnail generation, screencapturing, and adjustable-speed playback. It can be used for playback of long videos such as TV series on video platforms as well as portal development. For more information, see Overview.

#### Video copyright protection

Player supports video security capabilities of VOD, including private protocol encryption, offline download, scrolling text, and hotlink protection, to help you protect your video security. For more information, see Overview.



### Live recording

Player supports live recording playback, time shifting during live streaming, and pseudo-live streaming to help you deliver an integrated viewing experience in audio/video live and VOD playback scenarios.

### SDK Download

To try out the Player demo, see Free Demo.

To download the applicable SDK for integration, see SDK Download.

To learn more about the features and capabilities of Player, see Feature Description.

### Integration Guide

To help you quickly integrate Player, we provide a Player integration guide to describe the integration steps using demos.

If you encounter any playback issues, see FAQs.

### More Media SDKs

In addition to Player SDK, we also provide User Generated Short Video (UGSV), Player and Effect SDKs to quickly meet different applications requirement. You can choose the appropriate SDK you need. If you need both MLVB, UGSV, Player functions, you can also choose the All-In-One SDK<sub>o</sub>

Player SDK

MLVB SDK

UGSV SDK

TRTC SDK

All-In-One SDK



# **Basic Concepts**

Last updated : 2023-06-16 10:59:52

This document describes the basic concepts involved in the VOD playback scenarios of Tencent Player to help you quickly understand and use its VOD capabilities.

### Concepts

Player can be connected to Tencent Cloud VOD through Player or Player Adapter.

### Player

Player is a standalone and complete video player with a full range of video playback features, such as video encryption, thumbnail preview, and definition switch. It comes with complete UIs and demos, is deeply integrated with VOD, and can play back resources in VOD through their FileID. In addition, it provides a full-linkage VOD playback quality data service. To quickly connect Player, see Stage 1. Play back a video with player.

### **Player Adapter**

Player Adapter is a player plugin used to connect a third-party player with resources in VOD. It offers diverse capabilities, including video playback and encryption, so you can integrate it into your third-party players to play back resources in VOD through their FileID. To quickly connect Player Adapter, see the integration documents for different platforms.

### How to Select Player

To simplify connection and match your business scenarios, we recommend you select the most suitable player type when connecting the player service.



**Player**: Recommended for users who haven't integrated a player but need to quickly build VOD playback capabilities. Player is fully featured and easy to connect. VOD provides a detailed guide on how to connect it.

**Player Adapter**: Recommended for users who need to play back resources in VOD with a proprietary or third-party player. VOD offers Player Adapter to help you smoothly connect and use resources in VOD.

Platforms supported by Player include:

| Player Type | Player | Player Adapter |
|-------------|--------|----------------|
| Web         | 1      | $\checkmark$   |
| iOS         | 1      | $\checkmark$   |
| Android     | 1      | ✓              |
| Flutter     | 1      | -              |
| UI          | 1      | -              |
| Demo        | 1      | -              |

# Features

Last updated : 2024-06-28 17:41:57

The Player SDK provides video playback capabilities for live streaming and video-on-demand (VOD) on platforms such as web/HTML5, iOS, Android, and Flutter. Supported features are detailed below.

| FeatureFunctional ItemOverviewModuleItemItem |                        | Web  | iOS &<br>Android                 | Flutter   |                         |
|--|------------------------|--|----------------------------------|---|-------------------------|
| Playback<br>protocol/format                  | VOD/live<br>streaming  | Supports both<br>VOD and live<br>streaming   | √                                | √   | ✓                       |
|  | Live streaming formats | Supports<br>formats such as<br>RTMP, FLV,<br>HLS, DASH,<br>and WebRTC  | WebRTC,<br>FLV,HLS,<br>DASH      | RTMP,<br>FLV,HLS,<br>WebRTC   | RTMP,<br>FLV,HLS,WebRTC |
|  | VOD formats            | Supports audio<br>and video<br>formats such as<br>HLS, DASH,<br>MP4, and MP3   | HLS,<br>MP4,<br>MP3,FLV,<br>DASH | MP4, MP3,<br>HLS,<br>DASH<br>(DASH is<br>only<br>supported<br>in the<br>premium<br>version) | MP4,<br>MP3,HLS         |
|  | URL playback           | Supports URL<br>playback for<br>online videos,<br>where the URL<br>can be used for<br>either VOD or<br>live streaming            | ✓                                | ✓   | ✓                       |
|  | File ID playback       | Supports video<br>playback<br>through VOD<br>file identification<br>(FileID),<br>including videos<br>of multiple<br>resolutions, | ✓                                | ✓   | ✓                       |



|                            | thumbnails,<br>markers, and<br>other<br>information  |   |  |   |
|----------------------------|--|---|--|---|
| Local video<br>playback    | Supports<br>playback of<br>videos stored<br>locally  | -   | ✓  | ✓ |
| Live Event<br>Broadcasting | Supports<br>Tencent Cloud<br>Live Event<br>Broadcasting<br>playback with<br>millisecond-<br>level ultra-low<br>latency   |   | ✓  | ✓ |
| DASH protocol              | Supports DASH<br>video playback<br>with standard<br>protocols  | <b>√</b>  | <ul> <li>✓</li> <li>(supported<br/>only in the<br/>premium<br/>version)</li> </ul> | × |
| Panoramic VR<br>video      | Supports<br>playback of<br>panoramic VR<br>video sources.<br>Mobile devices<br>allow finger<br>dragging or<br>gyroscope<br>operations to<br>view panoramic<br>video content,<br>while PC<br>devices support<br>dragging with a<br>mouse to<br>animate and<br>view the<br>interface | ✓<br>(supported<br>only in the<br>premium<br>version) | ×  | × |
| QUIC<br>acceleration       | Supports the QUIC transport  | -   | 1  | ✓ |



|   | protocol,<br>effectively<br>improving video<br>transmission<br>efficiency                                    |                        | (supported<br>only in the<br>premium<br>version)                           |              |
|---|--|------------------------|--|--------------|
| SDR/HDR video<br>playback   | Supports<br>playback of<br>SDR videos<br>and HDR<br>videos in HDR<br>10/HLG<br>standards                     | -                      | √  | ✓            |
| H.264 video<br>playback and<br>software and<br>hardware<br>decoding | Supports<br>playback of<br>H.264 video<br>sources,<br>including<br>software and<br>hardware<br>decoding      | ✓                      | ✓  | ✓            |
| H.265 video<br>hardware<br>decoding                                 | 5 video<br>vare<br>ding<br>ding<br>Supports<br>hardware<br>decoding<br>playback of<br>H.265 video<br>sources |                        | ✓  | ✓            |
| AV1   | Supports<br>playback of<br>videos encoded<br>in AV1 format   | Partially<br>supported | Partially<br>supported<br>(supported<br>only in the<br>premium<br>version) | ×            |
| Audio playback  | Supports<br>playback of<br>pure audio files<br>such as MP3   | 1                      | J  | ✓            |
| Dual-channel<br>audio   | Supports<br>playback of<br>dual-channel<br>audio   | ×                      | $\checkmark$   | $\checkmark$ |



|                         | Multiple audio<br>tracks             | Supports<br>playback of<br>video files with<br>multiple audio<br>tracks, allowing<br>for switching<br>between tracks,<br>such as from<br>English to<br>Chinese   | ✓ | <ul> <li>✓</li> <li>(supported<br/>only in the<br/>premium<br/>version)</li> </ul> | ×            |
|-------------------------|--------------------------------------|--|---|--|--------------|
|                         | Http Header<br>Setting               | When<br>requesting<br>video<br>resources, you<br>can customize<br>HTTP Header<br>content   | × | ✓  | ✓            |
|                         | HTTPS                                | Supports<br>playback of<br>video resources<br>over HTTPS   | √ | ✓  | $\checkmark$ |
|                         | HTTP 2.0                             | Supports the<br>HTTP 2.0<br>protocol   | ✓ | √  | ✓            |
| Playback<br>performance | Short Video<br>Playback<br>Component | Achieve ultra-<br>fast first frame,<br>seamless start<br>playback, and<br>silky-smooth<br>short video<br>playback<br>experience at<br>an extremely<br>low cost of<br>integration. By<br>combining pre-<br>play, pre-<br>download,<br>player reuse,<br>precise traffic<br>control, and<br>loading | - | ✓<br>(supported<br>only in the<br>premium<br>version)                              | ×            |



|                           | strategies, this<br>ensures an<br>extremely<br>smooth<br>playback effect<br>with low energy<br>consumption  |   |                                      |                                 |
|---------------------------|---|---|--------------------------------------|---------------------------------|
| Pre-<br>downloading       | Supports pre-<br>downloading of<br>specified video<br>file content, and<br>configuration of<br>specified size<br>and resolution<br>for pre-<br>downloaded<br>video files.   | ✓ | ✓                                    | ✓                               |
| Streaming with caching    | Supports<br>content caching<br>during playback<br>to reduce<br>network usage.<br>A caching<br>policy can be<br>set  | ✓ | ✓                                    | ✓                               |
| Exact seek                | Exact seek |   | ✓                                    | ✓                               |
| Adaptive bitrate<br>(ABR) | Supports ABR<br>streaming of<br>HLS, DASH,<br>and WebRTC,   | 1 | ✓<br>(Only the<br>premium<br>version | ✓<br>(DASH is not<br>supported) |

|                                  | which can<br>automatically<br>select the<br>appropriate<br>bitrate for<br>playback based<br>on network<br>bandwidth   |   | supports<br>DASH)              |   |
|----------------------------------|---|---|--------------------------------|---|
| Real-time<br>download speed      | Provides real-<br>time download<br>speed<br>monitoring,<br>allowing display<br>to end-users<br>during buffering<br>as needed. It is<br>also crucial for<br>using the ABR<br>bandwidth<br>prediction<br>module | ✓ | ✓                              | ✓ |
| Multiple<br>instances            | Supports<br>adding multiple<br>players for<br>simultaneous<br>playback on a<br>single interface   | ✓ | √                              | ✓ |
| Dynamic frame<br>rate adjustment | When lag<br>occurs, a "fast<br>forward"<br>method can be<br>employed to<br>automatically<br>catch up to<br>maintain real-<br>time quality<br>during live<br>streaming   | ✓ | ×                              | × |
| PDT Seek                         | Jump to a<br>specific PDT in<br>the video<br>stream to  | x | ✓<br>(supported<br>only in the | 1 |



|                     |   | enable fast-<br>forward,<br>rewind, and<br>progress bar<br>jumps  |   | premium<br>version)                                   |   |
|---------------------|---|---|---|---|---|
| Playback<br>control | Basic controls                                  | Supports start,<br>end, pause, and<br>resume<br>playback<br>control features  | V | V   | ✓ |
|                     | Basic picture-in-<br>picture<br>component       | Supports<br>switching to<br>picture-in-<br>picture for<br>playback in a<br>small window,<br>with mobile<br>support for<br>picture-in-<br>picture<br>playback both<br>within and<br>outside the<br>integrated App. | ✓ | ✓   | ✓ |
|                     | Advanced<br>picture-in-<br>picture<br>component | Compared to<br>basic picture-in-<br>picture, it adds<br>support for<br>encrypted video<br>picture-in-<br>picture, offline<br>playback<br>picture-in-<br>picture, and the<br>"instant switch"<br>effect            | - | ✓<br>(supported<br>only in the<br>premium<br>version) | × |
|                     | Seek within caching                             | Supports not<br>clearing<br>Caching video<br>during seek and<br>allows for fast<br>seek   | ✓ | ✓   | ✓ |



| Live streaming<br>time shifting   | Supports live<br>streaming time<br>shifting<br>playback, with<br>settings of start,<br>end, and<br>current<br>positions and a<br>draggable<br>progress bar |   | × | × |
|---|--|---|---|---|
| Progress bar<br>marking and<br>thumbnail<br>preview   | Supports<br>adding marker<br>information on<br>the progress<br>bar and<br>supports Sprite<br>Thumbnail<br>previews   | ✓ | ✓ | ✓ |
| Cover   | Supports<br>setting the<br>video cover for<br>playback   | 1 | √ | ✓ |
| Replay, loop<br>playback, list<br>playback<br>k list<br>playback<br>k list<br>playback<br>k list<br>playback<br>k list<br>playback<br>k list<br>playing videos<br>in a list<br>sequentially,<br>and even<br>looping them by<br>automatically<br>playing the first<br>video again<br>after the last<br>one finishes. |  | ✓ | ✓ | ✓ |
| Breakpoint<br>resume  | Supports<br>playback from<br>the last stop<br>position   | ✓ | 1 | ✓ |



| Custom<br>playback start<br>time | Supports<br>custom video<br>start time   | ✓ | ✓ | ✓ |
|----------------------------------|--|---|---|---|
| Playback speed<br>change         | Supports 0.5–<br>3x speed<br>change for<br>playback, with<br>pitch-<br>preserving<br>audio speed<br>change     | ✓ | ✓ | ✓ |
| Background<br>playback           | Supports<br>continued audio<br>and video<br>playback when<br>the interface is<br>switched to the<br>background | - | ✓ | ✓ |
| Playback<br>callback             | Supports<br>callback for<br>playback<br>status, first<br>frame,<br>completion, or<br>failure of<br>playback    | ✓ | √ | ✓ |
| Retry upon<br>playback failure   | Automatic retry<br>upon playback<br>failure, includes<br>auto-<br>reconnection<br>during live<br>streaming     | ✓ | V | ✓ |
| Volume settings                  | Supports real-<br>time adjustment<br>of system<br>volume and<br>mute operations                                | ✓ | √ | ✓ |
| Resolution switching and         | Supports seamless and  | 1 | 1 | ✓ |



| naming                | buffer-free<br>switching of<br>HLS video<br>streams at<br>various<br>resolutions, and<br>allows for<br>custom naming<br>of different<br>resolution<br>streams    |   |   |   |
|-----------------------|--|---|---|---|
| Screen-<br>capturing  | Supports<br>intercepting any<br>frame of the<br>playback<br>screen   | - | ✓   | × |
| Preview               | Supports<br>playing videos<br>with the<br>preview feature<br>enabled   | ✓ | √   | x |
| On-screen<br>comments | Supports<br>displaying on-<br>screen<br>comments for<br>the video  | ✓ | √   | × |
| External<br>subtitles | Supports<br>importing<br>custom subtitle<br>files; Web<br>version<br>supports<br>WebVTT<br>format, while<br>mobile version<br>supports VTT<br>and SRT<br>formats |   | ✓<br>(supported<br>only in the<br>premium<br>version) | x |
| SEI Callback          | Parse SEI<br>frames in the<br>video stream   | × | ✓<br>(supported<br>only in the                        | × |





|                |                                | and perform event callbacks  |   | premium<br>version)                                   |   |
|----------------|--------------------------------|--|---|---|---|
|                | HEVC Degraded<br>Playback      | The player<br>supports the<br>simultaneous<br>input of HEVC<br>and other video<br>encoding<br>formats such<br>as: H.264<br>playback links.<br>When the<br>playback links.<br>When the<br>playback<br>device does not<br>support the<br>HEVC format, it<br>will<br>automatically<br>degrade to the<br>video playback<br>of the other<br>configured<br>encoding<br>formats (such<br>as: H.264) | × | ✓<br>(supported<br>only in the<br>premium<br>version) | × |
|                | Volume<br>Equalization         | Automatically<br>adjust the<br>volume during<br>audio playback<br>to ensure a<br>consistent<br>volume level<br>across all audio<br>tracks  | × | ✓<br>(supported<br>only in the<br>premium<br>version) | × |
| Video security | Referer<br>blocklist/allowlist | Supports<br>identifying the<br>request source<br>through the<br>Referer field in<br>the playback<br>request and<br>controlling<br>source requests  | ✓ | •   | ✓ |



|                           | through the<br>blocklist or<br>allowlist   |          |  |   |
|---------------------------|--|----------|--|---|
| Key hotlink<br>protection | Supports<br>adding control<br>parameters in<br>the playback<br>link to manage<br>link validity,<br>preview<br>duration, and<br>number of IPs<br>allowed for<br>playback  | ✓        | ✓  | ✓ |
| HLS encryption            | Supports AES<br>encryption<br>provided<br>through HLS by<br>using a key to<br>encrypt video<br>data  | ✓        | 1  | ✓ |
| HLS private<br>encryption | Supports<br>encrypting<br>videos in Cloud<br>VOD's private<br>protocol, which<br>can only be<br>decrypted and<br>played through<br>the Player SDK,<br>effectively<br>preventing the<br>cracking by<br>various browser<br>plugins and<br>gray-market<br>tools | ✓        | ✓  |   |
| Commercial-<br>grade DRM  | Provides native<br>encryption<br>solutions like<br>Apple FairPlay  | <i>J</i> | <ul> <li>✓</li> <li>(supported<br/>only in the<br/>premium<br/>version)</li> </ul> | × |



|                       | and Google<br>Widevine  |                                |   |   |
|-----------------------|---|--------------------------------|---|---|
| Secure<br>download    | Supports the<br>decryption and<br>playback of<br>offline<br>downloaded<br>encrypted<br>videos only<br>through the<br>Player SDK   | -                              | ✓ | ✓ |
| Dynamic<br>watermarks | Supports<br>adding dynamic<br>text watermarks<br>to the playback<br>interface to<br>effectively<br>prevent piracy   | <i>J</i>                       | ✓ | × |
| Ghost<br>watermarks   | Randomly<br>appear on the<br>playback<br>interface for a<br>short time at<br>random<br>positions, and<br>the video<br>playback will<br>automatically<br>pause if an<br>exceptional<br>removal of the<br>watermark is<br>detected. This<br>ensures video<br>security with<br>minimal impact<br>on viewing<br>experience. |                                | × | × |
| Web security plugins  | Checks the<br>Web playback<br>environment<br>and status and   | ✓<br>(supported<br>only in the | - | - |



|                |                      | pauses<br>exceptional<br>video playback<br>to protect video<br>security. The<br>plugins cover<br>MSE<br>environment<br>detection,<br>security<br>architecture<br>check, and<br>interface<br>response<br>integrity<br>verification. | premium<br>version) |    |   |
|----------------|----------------------|--|---------------------|----|---|
| Display effect | Custom UI            | The SDK offers<br>an integrated<br>solution with UI,<br>which provides<br>UI-included<br>common<br>playback<br>components<br>that can be<br>selected as<br>needed  | ✓                   | ✓  | ✓ |
|                | Screen filling       | Supports<br>choosing<br>different fill<br>patterns to fit<br>the video<br>screen size  | ✓                   | \$ | × |
|                | Player size settings | Supports customizing the player size   | $\checkmark$        | ✓  | ✓ |
|                | Image stickers       | Supports<br>adding image<br>stickers for<br>advertising<br>during playback<br>pause  | ✓                   | ✓  | × |



| Video mirroring          | Supports<br>mirroring in<br>horizontal,<br>vertical, and<br>other directions   | ✓ | 1 | × |
|--------------------------|--|---|---|---|
| Video rotation           | Supports<br>rotating video<br>images by<br>angle and<br>automatically<br>rotating videos<br>based on the<br>rotate<br>parameter in<br>the video file | × | ✓ | × |
| Screen locking           | Supports the<br>screen locking<br>feature,<br>including<br>locking rotation<br>and hiding<br>interface<br>elements                                   | - | ✓ | × |
| Brightness<br>adjustment | Supports<br>adjusting<br>system<br>brightness<br>during video<br>playback  | - | ✓ | ✓ |

#### Note:

"-" means the feature is not supported or the concept doesn't exist on the platform.

"" without "supported only in the premium version" means it is supported in the basic version.

## Free Demo

Last updated : 2024-04-11 16:11:38

The Player SDK demo provides complete product-grade interactive UIs and business source code for you to use on demand.

### Demo

### Demos

### Player for web (TCPlayer)

TCPlayer supports video playback in a browser on a PC or mobile device. It provides demo pages for you to compare and view the video playback features and code. You can modify the code samples to view the effect of modified features in the playback area in real time.



### Develop and Debug Demo

To help developers better understand how to use the Player SDK, the Player SDK mobile version offers a demo source code for development and debugging, along with instructions on interfacing. You can follow the steps below for usage.



### Step 1: Access the Demo project source code

You can visit the following GitHub address to access the demo source code for debugging, or download the corresponding ZIP package.

| Platform | Source code address | Download ZIP package |
|----------|---------------------|----------------------|
| iOS      | GitHub              | ZIP package          |
| Android  | GitHub              | ZIP package          |
| Flutter  | GitHub              | -                    |

### Step 2. Configure the license

The Player SDK for mobile (iOS & Android & Flutter) requires access to a License for use.

1. Sign in to the VOD console or CSS console, select License Management > Mobile License on the left menu, and click Create Trial License.





2. Fill in the App Name , Package Name , and Bundle ID according to your actual needs, check the feature

module Player Premium Version, and click Confirm.

Package Name: Please peek at the build.gradle file in the App directory for the applicationId.

Bundle ID: Please peek at the xcode for the project's Bundle Identifier.

#### Note:

If the Package Name or Bundle ID applied for in the Tencent Cloud Console is inconsistent with the actual Package Name or Bundle ID in the project, playback will fail.

| Create trial li | cense                             |   |                               |                     | >            |
|-----------------|-----------------------------------|---|-------------------------------|---------------------|--------------|
| Basic inform    | ation                             |   |                               |                     |              |
| i App n         | ame, Package                      | name, and Bundle ID an                                | e required and (              | can be modified I   | ater.        |
| App name        |                                   |   |                               |                     |              |
|                 | Max 128 by<br>underscore          | tes; supports letters, Ch<br>s, hyphens, and periods. | inese character<br>E.g.: TRTC | rs, numbers, spac   | :es,         |
| Package name    |                                   |   |                               |                     |              |
|                 | Max 128 by<br>periods. E.ç        | tes; supports letters, nu<br>: tencent.trtc.com       | nbers, spaces,                | underscores, hyp    | ohens, and   |
| Bundle ID       |                                   |   |                               |                     |              |
|                 | Max 128 by<br>periods. E.ç        | tes; supports letters, nur<br>.: tencent.trtc.com     | nbers, spaces,                | underscores, hyp    | phens, and   |
| Capability      |                                   |   |                               |                     |              |
| i Each days a   | trial license ca<br>and cannot be | n only be used for one ca<br>renewed after expiration | apability. A trial            | license is valid fo | r 28         |
| UGSV Star       | ndard                             |   | N                             | Valid for 28 days   | Already used |
| MLVB            |                                   |   | 1                             | Valid for 28 days   | Already used |
|                 |                                   |   |                               |                     |              |
| Player Prei     | mium                              |   | `                             | Valid for 28 days   | Available    |

3. After the trial version License is successfully created, the page will display the generated License information. Upon SDK initialization, the License URL and License Key are required; please save the following information carefully.



| 11 Trial license<br>Package name 111 | Bundle ID 111 Creation time 2024-03-26 12:05-42 | Edit |
|--------------------------------------|---|------|
| Basic informati                      |   |      |
| Player Premiun                       | n   |      |
| Status                               | Normal  |      |
| Feature                              | Video Playback Premium                          |      |
| Validity period                      | 2024-03-26 12:05:42 - 2024-04-24 00:00:60       |      |
|                                      |   |      |

4. After accessing the Licence URL and Licence Key, please refer to the tutorial below to configure them in the Demo project.

Android configuration for Licence

iOS configuration for Licence

Flutter configuration for Licence

Open the Demo/app/src/main/java/com/tencent/liteav/demo/TXCSDKService.java file, and replace the Licence URL and License Key with the applied Licence content.

| C TXCSD | KService.java ×   |
|---------|---|
|         |   |
|         | <pre>public class TXCSDKService {</pre>   |
|         | <pre>private static final String TAG = "TXCSDKService";</pre>                       |
|         | // 如何获取License? 请参考官网指引 <u>https://cloud.tencent.com/document/product/454/34750</u> |
|         | <pre>private static final String licenceUrl =</pre>                                 |
|         | "请替换成您的licenseUrl";   |
|         | <b>private static final String l<i>icenseKey</i> =</b> "请替换成您的licenseKey";          |

Open the Demo/TXLiteAVDemo/App/config/Player.plist file, and replace the Licence URL and Licence Key with the applied Licence content.

|                           | B I < > ■ Player  | ₹≣0   (         |
|---------------------------|---|-----------------|
| V 🖾 TXLiteAVDemo          | $\fbox{TXLiteAVDemo} > \fbox{TXLiteAVDemo} > \fbox{App} > "\raised config > \raised Player > \raised config > \raised Player > \raised config $ | < 🔺             |
| ✓                         | 1 xml ·version="1.0" ·encoding="UTF-8"? ¬   |                 |
| > 🚞 Resources             | 2 'plist 'PUBLIC ''-//Apple//DTD 'PLIST '1.0//EN" '"http://www.apple.com/DTDs/PropertyList-1.0.dtd</p   | <b>!"&gt;</b> ¬ |
| > 🚞 SuperPlayFeedDemo     | 3 <plist version="1.0">¬</plist>  |                 |
| > 🚞 UGCShortVideoPlayDemo | 4 <dict>¬</dict>  |                 |
| ~ 🚞 Арр                   | 5 » <key>licenceConfig</key> ¬  |                 |
| ~ 🚞 config                | 6 » <dict>¬</dict>  |                 |
| Player                    | 7 » » <kev>licenceKev</kev> ⊐   |                 |
| > TRTCInfoPlist           | 8 » » (string>]   |                 |
| > AppCommon               | 9 》 》   |                 |
| > Main                    |   |                 |
| > Resource                |   |                 |

Open the Flutter/example/lib/main.dart file, and replace the Licence URL and Licence Key with the applied Licence content.

| 【 main.dart × |  |  |
|---------------|--|--|
| 37 🛱          | }  |  |
| 38            |  |  |
| 39            | /// set player license   |  |
| 40 🕂          | Future <void> initPlayerLicense() async{</void>                              |  |
| 41            | <b>String licenceURL =</b> ""; // 获取到的 licence url                           |  |
| 42            | <b>String licenceKey =</b> "" <b>;</b> // 获取到的 licence key                   |  |
| 43            | <pre>await SuperPlayerPlugin.setGlobalLicense(licenceURL, licenceKey);</pre> |  |
| 44            | }  |  |
|               |  |  |

# Free Trial License

Last updated : 2024-04-11 16:11:37

The Player SDK offers a trial License, including **Mobile** trial License and **Web** trial License. You can refer to this document for guidance and apply for a free trial as needed.

| Trial License Types  | Features that can be authorized                                    | Validity Period  |
|----------------------|--|--|
| Mobile trial License | Player SDK <b>Mobile</b> all features (including premium features) | By default, 28 days after application  |
| Web trial License    | Player SDK <b>Web</b> all features (including premium features)    | By default, 14 days after application, renewable once for a total of 28 days |

### Note:

After purchase, you can add and renew the player License via the VOD console or CSS console. For details, please see Adding and Renewing License.

### Mobile trial License

### **Trial License Application**

You can apply for a free trial of the Player Mobile License (valid for total of 28 days) for testing. When applying for the test module, you can choose to **create a new trial License and select Player trial License** or **apply for a new License in an existing test application**.

Method 1: Create a new trial License and select Player trial License

Method 2: Apply for a New Trial License within an Already Created Test Application

1. Sign in VOD console or CSS console> License Management > Mobile Client License, click Create Trial License.



2. Fill in the App Name, Package Name, and Bundle ID according to actual needs, select Player Premium, and click Create.

| Basic inform   | ation  |   |                                      |  |   |
|--|--|---|--------------------------------------|--|---|
| <ol> <li>App na</li> </ol>                           | ame, Package na                                  | ame, and Bundle                         | ID are required a                    | and can be modified  | later.                                    |
| App name   |  |   |                                      |  |   |
|  | Max 128 bytes<br>underscores, I                  | s; supports letter<br>hyphens, and per  | s, Chinese chara<br>iods. E.g.: TRTC | acters, numbers, spac<br>;   | ces,                                      |
| Package name   |  |   |                                      |  |   |
|  | Max 128 bytes<br>periods. E.g.:                  | s; supports letter<br>tencent.trtc.com  | s, numbers, spa                      | ces, underscores, hy   | phens, and                                |
| Bundle ID  |  |   |                                      |  |   |
|  | Max 128 bytes<br>periods. E.g.:                  | s; supports letter<br>tencent.trtc.com  | s, numbers, spa                      | ces, underscores, hy   | phens, and                                |
| Capability   |  |   |                                      |  |   |
| -  |  |   |                                      |  |   |
| <ul> <li>Each t<br/>days a</li> </ul>                | rial license can o<br>Ind cannot be rer          | only be used for o<br>newed after expir | ne capability. A<br>ation.           | trial license is valid fo  | or 28                                     |
| (i) Each t<br>days a<br>UGSV Stan                    | rial license can o<br>Ind cannot be rer<br>dard  | only be used for c<br>newed after expir | ne capability. A<br>ation.           | trial license is valid fo  | or 28<br>Already used                     |
| Each t<br>days a<br>UGSV Stan<br>MLVB                | rial license can o<br>Ind cannot be rer<br>Idard | only be used for c<br>newed after expir | ne capability. A<br>ation.           | trial license is valid fo<br>Valid for 28 days<br>Valid for 28 days                      | Already used                              |
| Each t<br>days a<br>UGSV Stan<br>MLVB<br>Player Prer | rial license can o<br>Ind cannot be rer<br>Idard | only be used for c                      | ne capability. A<br>ation.           | trial license is valid fo<br>Valid for 28 days<br>Valid for 28 days<br>Valid for 28 days | Already used<br>Already used<br>Available |

3. After the trial License is successfully created, the page will display the generated License information. When initializing the SDK configuration, you need to input both the Key and License URL. Please save the following information accordingly.

#### Note:

For the same application, the License URL and Key are unique. When the trial License upgrades to an official version, the License URL and Key remain unchanged.

| Trial license                                   | Evends ID 111 Creation time 2024-03-28 12:05-42 | Ed |
|---|---|----|
| Basic information<br>License URL<br>License key |   |    |
| Player Premium                                  |   |    |
| Feature   | Voldoo Plavback Premium                         |    |
| Validity period                                 | 2024-03-26 12:06:42 - 2024-04-24 00:00:00       |    |
|   |   |    |

If you wish to apply for a trial of the **Player** feature within an already created test application, follow these steps: 1. Sign in to VOD console or CSS console> License Management > Mobile License, select the application you wish to test, and click **Test New Feature**.

| Package name 141                 | 12213 Bundle ID 14123123 Creation time 2024-  | 4-03-01 11:19:24 |                       |
|----------------------------------|---|------------------|-----------------------|
| Basic inform                     | ation   |                  |                       |
| License URL                      |   |                  |                       |
| License key                      | CONTRACTOR AND  |                  |                       |
|                                  |   |                  |                       |
|                                  |   |                  |                       |
| UGSV Standa                      | ard   |                  |                       |
| UGSV Standa                      | ard Expires in 4 day(s), 4 hour(s), and 51 minute(s)  |                  | Toy more canabilities |
| UGSV Standa<br>Status<br>Feature | ard<br>Expires in 4 day(s), 4 hour(s), and 51 minute(s)<br>Video production (standard) + Video playback |                  | Try more capabilities |

2. Select **Player Premium**, and click **OK**.

| ()    | Each trial license can<br>days and cannot be r | only be used for one capat<br>renewed after expiration. | bility. A trial license is valid for 28 |         |
|-------|--|---|---|---------|
| UGS   | V Standard                                     |   | Valid for 28 days Alread                | dy used |
| MLVI  | B  |   | Valid for 28 days Alread                | dy used |
| Playe | er Premium                                     |   | Valid for 28 days Availa                | ble     |

#### Note

During the trial License's validity period, you can click on the **Edit** button on the right, enter to modify the Bundle ID and Package Name information, and click **OK** to save.

If there is no Package Name or Bundle ID, you may fill in "-".

### Web Trial License

### **Trial License Application**

You can apply for a free Player Web Client trial License (valid for 14 days, renewable once, totaling 28 days) for a test experience.

1. Sign in to VOD console or CSS console> License Management >, click Create Trial License.





2. Fill in the Project Name and Domain, then click Create to complete the application.

| Create Playe              | er trial license   | × |
|---------------------------|--|---|
| i Playe<br>renew<br>Playe | r trial license only supports one application with 14 days validity period and one<br>val chance, totally 28 days. Trial license supports all features included in Web<br>r Premium. |   |
| Project name              | test<br>Support English, Chinese, number, space, _, -, ., up to a maximum of 128 bytes   |   |
| Domain                    | *.test.com<br>Only wildcard domain name bound is supported, up to a maximum of 128<br>bytes<br>One Web premium license can bind one wildcard domain name.                            |   |
|                           | Create Cancel  |   |

#### Note:

Only version 5.1.0 and above of the Web Player SDKsupport the use of wildcard domain authorization.

3. After the trial License is successfully created, the page will display the generated License information. When initializing the SDK configuration, you need to input both the Key and License URL. Please save the following information accordingly.



| Creation time 2024-03-25 15:12:53 |                 |   |       |
|-----------------------------------|-----------------|---|-------|
| Authorization Information         | Player Premium  |   | Renev |
| License URL                       | Status          | Normal                                    |       |
| License key                       | Feature         | Web Video Playback Premium                |       |
|                                   | Validity period | 2024-03-25 19:11:53 - 2024-04-09 00:00:00 |       |
| Domain Information                |                 |   |       |
| domain *.test.com                 |                 |   |       |

#### Note:

A single Web Premium Version License can only be associated with one wildcard domain.

You can click the Edit on the right side to modify the domain name.

### Trial License Renewal

1. The Web trial License initially has a default validity period of 14 days. You can renew it **once**. Click the **Player Premium** feature on the right and select **Renew**.

| ▼ test Trial license<br>Creation time 2024-03-25 15:12:53 |   | Edit  |
|---|---|-------|
| Authorization Information                                 | Player Premium           Status         Normal           Feature         Web Video Playback Premium           Validity period         2024-03-25 19:11:53 - 2024-04-09 00:00:00 | Renew |
| Domain Information<br>domain *test.com                    |   |       |

2. Click Ensure for renewal to renew the feature for 14 days.



Note:



The trial License has a total validity of 28 days, and **can only be renewed once**. If you wish to continue using it, please purchase and bind Web Formal License.
# Purchase Guide Billing Overview

Last updated : 2024-04-11 16:11:38

#### The cost of the Player SDK includes the following:

| Type of Expense                  | Description  |
|----------------------------------|--|
| SDK authorization fees           | The use of authorization fees for Player SDK.  |
| Other related cloud service fees | When using the Player SDK in conjunction with cloud services like VOD and CSS, corresponding fees are generated. |

#### Note:

The Player SDK offers a cumulative 28-day trial License for both the Mobile Premium Version and the Web Premium Version. You can apply for free in the console.

## **SDK Authorization Fees**

The Player SDK authorizations for Mobile and Web are separate. Both platforms offer Basic and Premium versions, and you can obtain the usage authorization for the corresponding version by purchasing the designated License. For differences between the platforms and the versions, see Product Features.

## Acquisition Method

| Player SDK<br>Platform          | Feature<br>Version | Required License Type   | Price   | Authorization Unit  |
|---------------------------------|--------------------|---|---|---|
| Mobile Player<br>(iOS & Version | Basic<br>version   | Player Mobile Basic<br>Version License /UGSV<br>License /MLVB License | 0 USD (1 year,<br>renewable for free)<br>Apply for free | One license can<br>authorize one iOS<br>application Bundle                    |
| Flutter)                        | Premium<br>version | Player Mobile Premium<br>Version License                              | 499 USD/month<br>Purchase Now                           | application<br>Package Name   |
| Web Player                      | Basic<br>version   | Player Web Basic<br>Version License                                   | 0 USD (1 year,<br>renewable for free)<br>Apply for free | Precise domain<br>(one License can<br>authorize up to ten<br>precise domains) |
|                                 | Premium            | Player Web Premium  | 99 USD/month  | Wildcard Domain   |



|--|

If you have already purchased one of the following three Licenses, you can also obtain the Player Mobile SDK authorization with the Basic version without needing to purchase an additional player License.

| Official License Type      | Validity Period                    | Price (USD/Year) |
|----------------------------|------------------------------------|------------------|
| UGSV Lite Version License  | 1 Year (From the Date of Purchase) | 1,899            |
| UGSV Basic Version License | 1 Year (From the Date of Purchase) | 9,999            |
| MLVB License               | 1 Year (From the Date of Purchase) | 5,988            |

#### Billing

Each account can apply for one free mobile player trial license and one web player trial license via the VOD console or CSS console to experience and test the product. The first application provides a 14-day validity period, which can be renewed in the console for another 14 days, totaling 28 days. Mobile and web trial licenses have separate application counts and durations.

A mobile license can be associated with one iOS application Bundle ID and one Android application Package Name, while a web license can be associated with up to ten precise domains. Associating a license with an application/domain provides the corresponding authorization without distinction between business environments. If multiple applications need to be integrated, the corresponding number of licenses must be purchased for binding. Domain Name Description:

A precise domain refers to a specific fixed domain address, such as: a.com , a.b.com , a.b.c.com and other unique and fixed domain addresses.

A wildcard domain refers to a series of domain addresses with the same domain suffix, such as a wildcard domain \*.a.com associated with a license. It allows a.a.com, b.a.com, c.a.com and other domains with the same suffix to unlock package capabilities. The \* position can support custom definitions of multi-level domains, such as b.c.a.com and b.c.d.a.com.

An application supports changing the associated license. The authorization validity period for changing the application is the validity period of the newly associated license. The replaced license can be used to associate with other applications, and its validity period remains unchanged.

For refund-related content, see Refund Explanation.

### New License Validity Period Example

A customer purchased the Player Mobile Premium Version License A on July 1, 2022, at 11:36:59, and associated it with iOS application A1 and Android application A2. Later, on July 2, 2022, at 11:36:59, the customer purchased



another Player Mobile Premium Version License, obtained its License B, and associated it with iOS application B1 and Android application B2. Therefore:

The validity period of Player Mobile Premium Version License A is from July 1, 2022, 11:36:59 to August 2, 2022, 00:00:00.

The validity period of Player Mobile Premium Version License B is from July 2, 2022, 11:36:59 to August 3, 2022, 00:00:00.

Applications A1, A2, B1, and B2 all receive authorization to use features of the Player SDK Mobile Premium Version. Meanwhile, the authorization validity period for applications A1 and A2 is based on the validity of License A, and for applications B1 and B2, it is based on the validity of License B.

### Validity Period Update Example

Based on the previous example, the same customer purchased the Player Mobile Premium Version License C on December 1, 2022, 15:48:12, and used License C to renew applications A1 and A2. As a result:

The validity period of Player Mobile Premium Version License C is from August 1, 2022, 15:48:12 to September 2, 2022, 00:00:00. When renewing for applications A1 and A2, License C will replace the originally affiliated License A with applications A1 and A2.

Since an application's authorization validity period is based on the actual affiliated License's validity period, the renewal changes the validity period of applications A1 and A2 to that of License C, which is from August 1, 2022, 15:48:12 to September 2, 2022, 00:00:00. Thus, the authorization validity period of applications A1 and A2 is extended from the original August 2, 2022, 00:00:00 to September 2, 2022, 00:00:00.

After License C becomes affiliated with applications A1 and A2, the originally affiliated License A with applications A1 and A2 will naturally disaffiliate, while the validity period of License A remains unchanged, still serving as an available License resource that can be affiliated with other applications.

## Other Related Cloud Services Fee

In addition to license fees, using the Player SDK may also incur the following service fees. Related fees will not be generated if the related services are not used.

## Video on Demand (VOD)

Tencent Cloud's VOD service provides functions such as video upload, storage, transcoding, acceleration distribution playback, copyright protection, and play quality monitoring.

Billing of VOD:

Storage fees are charged based on the storage space used by files uploaded to VOD and their transcoding outputs.



If you transcode files stored in VOD, transcoding fees are charged based on the specifications and durations of the outputs.

If you use VOD's acceleration service to deliver videos, acceleration fees will be charged based on the traffic consumed for playback.

#### Note

For more information about the billing of VOD, see Billing Overview.

## Cloud Streaming Services (CSS)

CSS offers capabilities including live stream receiving, on-cloud recording, live transcoding, live screencapture, and live stream delivery and playback. If you need to use related functions, we recommend you use CSS.

Using CSS to receive and deliver live streams will incur basic service fees, which are charged based on the traffic/bandwidth consumed.

Using CSS features such as live transcoding (including stream mixing and watermarking), live recording, live screencapture, RTC-based co-anchoring, and relay to CDN will incur value-added service fees.

#### Note

For more information about the billing of CSS, see Pricing Overview.

# **Refund Policies**

Last updated : 2024-04-11 16:11:38

## Five-Day Unconditional Refund

To facilitate the use of the Player SDK, if your purchased License meets the following conditions, Tencent Cloud supports an unconditional refund within 5 days:

Mobile License not bound to an application package name (including both new additions and renewals binding methods), and Web License not bound to a domain.

The purchase date of the License is no more than 5 days ago (including the 5th day).

## **Refund Policy**

Orders that meet the Five-Day Unconditional Refund policy can submit a refund request via ticket, see Refund Steps for details.

If there is suspected abnormal or malicious return, Tencent Cloud reserves the right to reject your return request.

For orders that meet the 5-day unconditional refund policy, the refund amount will be the total payment made at the time of purchase, including the cash account balance, transferred earnings, and gift account balance.

#### Note

Rebates and vouchers will not be returned.

A full refund will be returned to your Tencent Cloud account.

## **Refund Steps**

- 1. Click Submit Ticket to go to the ticket submission page.
- 2. In the search box on the right, search for and select "Other Tencent Cloud Products".
- 3. Select the problem type as "Feature Consultation".
- 4. Go to the ticket creation page, fill in the relevant information, and click **Submit Ticket** to complete.

#### Note

Before returning the resources, please check if they meet the conditions for a 5-day unconditional refund. If they do not, a refund cannot be processed.

# **Overdue and Suspension Policy**

Last updated : 2024-04-11 16:18:08

When the Tencent Cloud billing platform detects that your Tencent Cloud account balance is insufficient, it will remind you to top up. You have 24 hours to replenish your account after it falls into arrears. If you do not top up your account after this period, services such as Cloud Streaming Services, Video on Demand, and live co-anchoring will be temporarily suspended. If you top up your account within 24 hours of falling into arrears, your co-anchoring service will not be affected.

# SDK Download SDK Download

Last updated : 2024-04-11 16:11:38

Player SDK

# Release Notes(Web)

Last updated : 2024-04-11 16:11:38

## TCplayer update log

## TCplayer 5.1.0 @ 2023.11.13

New support for configuration native decoding strategy MacOS iOS optimize subtitle display effect Full scene support for loadVideoByID method Support for parsing wildcard domain names Added new server-side error codes and license-related error codes Fixes

## TCplayer 5.0.0 @ 2023.08.16 Breaking Change

Adding a License.
New support for VR and security environment check plugins.
Live Event Broadcasting ABR interface adjustment.
Live Event Broadcasting downgrade logic adjustment, added automatic downgrade.
Open M4A.
Optimized pseudo-fullscreen effect.
macOS Safari supports private encryption.

### TCplayer 4.8.0 @ 2023.04.20

New support for P2P. Added a switch for downgrading WebRTC. Fixes.

## TCplayer 4.7.2 @ 2023.1.10

Fixed an issue with incorrect iOS environment detection.

## TCplayer 4.7.0 @ 2022.12.20

Added automatic asynchronous loading of dependency SDK.

Adding Element (XML) checks.

New support for ghost watermark.

Improving code aliasing.

## TCplayer 4.6.0 @ 2022.11.04

New support for multiple audio tracks. New support for URL-based resuming playback. Fixed some issues.

### TCplayer 4.5.4 @ 2022.08.26

New support for Live Event Broadcasting ABR, updated txliveplayer. New support for AV1 in FLV, updated flv.js. New support for SDMC DRM. New support for unifying the auto-play blocked events in Live Event Broadcasting and Live Video Broadcasting to 'blocked'. New support for marker callback events.

Fixed some issues.

### TCplayer 4.5.3 @ 2022.06.15

New support for commercial-grade VOD DRM.

### TCplayer 4.5.2 @ 2022.04.15

Fixed a vulnerability where videos could be stolen by hijacking MSE to bypass private encryption schemes. Fixed some data reporting issues.

### TCplayer 4.5.0 @ 2022.01.14

New support for data reporting features in Live Video Broadcasting and Live Event Broadcasting. New support for playing MP3 audio formats. New support for frame synchronization capabilities under poor network conditions. Optimized watermark feature, supporting more configuration options. Fixed several issues.

### TCplayer 4.4.0 @ 2021.12.14

New support for playing Live Event Broadcasting. New data reporting feature for on-demand scenarios. New support for playing original and transcoded media files via the v4 interface.

## TCplayer 4.1 @ 2020.07.10

Updated the default version of hls.js to 0.13.2. Support for enabling the Key Hotlink Protection feature. Fixes for other known issues have been implemented.

## TCplayer 4.0 @ 2020.06.17

Fix preview video to keep displaying the original duration. Enable background quality configuration. Fixes for other known issues have been implemented.

## TCplayerLite update log

## TCplayerLite 2.4.1 @ 2021.06.25

New support for WebRTC stream addresses with v1 signaling. Added webrtcConfig parameter. Added events for WebRTC stutter, stutter end, and stream end.

## TCplayerLite 2.4.0 @ 2021.06.03

Added support for the Live Event Broadcasting feature. Fixes for other known issues have been implemented.

## TCplayerLite 2.3.3 @ 2020.07.01

Fixed an issue where switching to fullscreen in the X5 environment causes abnormal event dispatch.

Avoided the issue where switching HLS sources causes related events to trigger slowly, leading to abnormal cover display.

## TCplayerLite 2.3.2 @ 2019.08.20

Updated the default HLS version to 0.12.4. Fixes for other known issues have been implemented.

## TCplayerLite 2.3.1 @ 2019.04.26

Added fivConfig parameter. Load flv.1.5.js by default. Fixes for other known issues have been implemented.

## TCplayerLite 2.3.0 @ 2019.04.19

Added some feature parameter options. Changed the parameter 'coverpic' to 'poster'. Destroy the flv.js instance. Fixes for other known issues have been implemented.

## TCplayerLite 2.2.3 @ 2018.12.17

Optimized playback logic.

Resolved the issue where loading animation appears on iOS WeChat without triggering play events. Fixes for other known issues have been implemented.

## TCplayerLite 2.2.2 @ 2018.05.03

Optimized the loading component. Optimized the Flash destroy method. Use H5 playback by default. Fixed known issues.

## TCplayerLite 2.2.1 @ 2017.12.20

Added configurable definition text feature. Set default definition. Supports method to switch definition.

## TCplayerLite 2.2.1 @ 2017.12.07

Added systemFullscreen parameter. Added flashUrl parameter. Fixed the UI issue when toggling mute at maximum volume. Fixed the issue where two clicks are required to play on iOS 11 WeChat. Fixed the issue where system styles are obscured in Safari 11. Adapted to the situation where seeking is triggered in the X5 kernel, but seeked is not. Fixed the problem where dragging the progress bar to the starting position and setting currentTime fails. Switching definition keeps the volume unchanged. Fixed the issue where the page width is 0, causing the player width determination to fail. The destroy method now completely removes the player node.

## TCplayerLite 2.2.0 @ 2017.06.30

Added parameters to control the playback environment: Flash, h5\_flv, x5\_player.

Adjusted player initialization logic to optimize error message effects.

Added support for flv.js, allowing FLV playback with flv.js under the right conditions.

Supports the x5-video-orientation attribute.

Enhanced the logic for determining the playback environment. Parameters can now adjust the priority between H5 and Flash, and whether to enable TBS playback.

Implemented version numbering for releases to avoid impacting users of older versions.

Optimized the timestamp for event triggers, standardizing it to universal time.

Bug fixes.

## TCplayerLite 2.1.0 @ 2017.03.04

As of 2017.06.30, after several iterations of development, it has gradually become stable. Currently, unless otherwise specified in the documentation, all feature descriptions are based on this version.

## Adapter plugin update log

## Adapter plugin released @ 2021.07.16

First release of the player Adapter version

# Release Notes(iOS & Android)

Last updated : 2024-08-07 15:13:59

## Player SDK

### Player SDK 12.0 @ 2024.08.01

#### New features:

Android&iOS: When HEVC is downgraded, the corresponding event

VOD\_PLAY\_EVT\_HEVC\_DOWNGRADE\_PLAYBACK (2031) is thrown externally.

Android&iOS: Add the event VOD\_PLAY\_EVT\_VOD\_PLAY\_FIRST\_VIDEO\_PACKET (2017) for receiving the first frame packet.

#### **Bug fixes:**

Android&iOS: Fixed the cache reuse error caused by the playback link with exper when MP4 preview video switches to the main film playback scene.

Android&iOS: Fixed the abnormal preloading and adaptive playback of HLS encrypted video.

Android&iOS: Fixed the memory leak problem that may occur in high-concurrency playback scenarios.

iOS: Fixed the problem that multiple pre-downloads initiated by fileId may cause crashes.

iOS: Fixed the problem that the app\_version field is missing in the playback quality monitoring report.

iOS: Fixed the problem that the error code is lost after selecting the subtitle track.

iOS: Fixed the issue that the subtitle data is lost after restarting the playback after adding subtitles after calling TXVodPlayer#stopPlay.

iOS: Fixed the issue that the playback starts from the beginning after returning to the foreground after going back to the background for a while during playback.

iOS: Fixed the issue that the memory is not released after the pre-download fails.

### Player SDK 11.9 @ 2024.06.03

#### **Bug fixes:**

Android&iOS: Optimize the first frame time statistics to make the statistics more accurate.

Android&iOS: Fix the crash problem of the player downloader module in multi-threaded scenarios.

Android: Remove the call to NetworkInfo#getExtraInfo to avoid being mistakenly detected as reading ssid.

#### Player SDK 11.8 @ 2024.05.06

#### New features:

Android&iOS: Support HLS EVENT live source (supported by the advanced version). Android: Support playing local videos with content:// and asset:// URIs.

Android&iOS: Support precise and non-precise Seek.



#### Bug fixes:

Android: Fix the problem of callback progress after receiving the PLAY\_EVT\_VOD\_PLAY\_PREPARE event when the TXVodPlayer#setAutoPlay parameter is false.

iOS: Fix the abnormal problem caused by directly stopPlay without calling exitPictureInPicture during picture-inpicture playback.

#### Player SDK 11.7 @ 2024.03.04

#### New features:

Android&iOS: Added playback volume equalization control function (supported by premium version).

Android&iOS: HECV adaptive downgrade playback (supported by premium version).

Android&iOS: Support HLS EVENT live broadcast source (supported by premium version).

#### Player SDK 11.6 @ 2024.01.10

#### New features:

Android & iOS: Upgraded network kernel for advanced player version, providing better performance (supported by premium version).

Android & iOS: Preloading supports FileId encrypted videos (supported by premium version).

Android & iOS: FileId video playback supports ghost watermark.

Android & iOS: Supports SEI information callback (supported by premium version).

Android & iOS: Supports real-time acquisition and seek ability of Program Date Time for HLS video format (supported by premium version).

#### Function optimization:

Android & iOS: First frame event carries additional first frame duration information.

Android & iOS: Fixed built-in subtitle parsing exception for advanced player version.

Android & iOS: No need to set http proxy to bypass localhost when using packet capture tool.

#### **Bug fixes:**

Android & iOS: Fixed issue where playback retry count is invalid after network disconnection.

Android & iOS: Fixed slow seek issue for some mp3 files.

Android & iOS: Fixed issue where AES-128 encrypted m3u8 files cannot be played offline.

Android: Fixed issue where only the first seek is effective when seeking multiple times in a short period.

iOS: Fixed issue where built-in subtitle causes abnormal playback in Picture-in-Picture mode.

#### Player SDK 11.4 @ 2023.08.29

#### **New Features:**

iOS: Added advanced picture-in-picture capabilities, supporting picture-in-picture playback of encrypted videos, offline playback of picture-in-picture, and automatic picture-in-picture when switching to the background.

#### Function Optimizations:



Android & iOS: Cache-related interfaces support KB granularity control.

Android & iOS: Optimized the network scheduling strategy of the player.

#### **Bug Fixes:**

Android & iOS: Fixed the issue where network traffic consumption could not be stopped in a timely manner after playback stopped.

#### Player SDK 11.3 @ 2023.07.07

#### New Features:

Android&iOS: Video preloading now supports specifying media types (TXPlayInfoParams#mMediaType) to reduce type detection and improve download efficiency.

iOS: Added network exception retry mechanism during playback.

#### **Function Optimizations:**

Android&iOS: Optimized the issue of excessive memory allocation during playback in the event of network disconnection.

#### **Bug Fixes:**

Android&iOS: Fixed issue where playing HLS videos after network disconnection caused repeated playback of a certain segment.

Android&iOS: Fixed issue where small preloadSize settings caused occasional video playback failures.

iOS: Fixed issue where httpDns service occasionally caused crashes in weak networks.

### Player SDK 11.2 @ 2023.06.01

#### New features:

Android&iOS: Added callback for VOD\_PLAY\_EVT\_HIT\_CACHE event during VOD playback.

Android: DRM playback can now be configured with COM or CN certificate providers using

TXPlayerGlobalSetting#setDrmProvisionEnv.

#### **Function optimizations:**

Android&iOS: Media type can now be specified using TXVodPlayConfig#setMediaType to reduce underlying type detection and improve startup speed.

Android&iOS: Added callback for audio bitrate (VIDEO\_BITRATE&AUDIO\_BITRATE) during MP4 playback.

#### **Bug fixes:**

Android&iOS: Fixed missing VIDEO\_CACHE value in onNetStatus callback.

Android&iOS: Fixed playback failure issue for offline downloaded resources in certain scenarios.

Android&iOS: Fixed probability of playback failure during fast video switching.

### Player SDK 11.1 @ 2023.04.07

#### **Function optimizations:**

Android&iOS: Added interface for checking if downloaded video resources exist (TXVodDownloadMediaInfo#isResourceBroken).



Android&iOS: Video downloading now supports sharing cache for private encrypted videos before the anti-theft link expires.

#### **Bug fixes:**

Android&iOS: Fixed issue where CPU usage increased after audio frame decoding failure.

Android&iOS: Fixed issue where progress was lost when the download process exited abnormally.

Android&iOS: Fixed playback issues for some MP4 files.

#### Player SDK 11.0 @ 2023.02.24

#### **Function optimizations:**

Android&iOS: Improved compatibility for video playback when audio and video are not well interleaved.

iOS: Optimized memory release for video frame data.

#### Bug fixes:

iOS: Fixed abnormal event of not calling back VOD\_PLAY\_EVT\_LOOP\_ONCE\_COMPLETE (single loop playback completed) when playing in a loop.

Android&iOS: Fixed inaccurate size of sub-stream files obtained through video downloading

(TXVodDownloadMediaInfo#getSize).

Android&iOS: Fixed abnormal download issue for nested m3u8 single-stream downloads.

Android&iOS: Fixed probability of playback issues for nested m3u8 private encrypted videos downloaded through video downloading.

#### Player SDK 10.9 @ 2022.12.30

#### New features:

Android&iOS: Support external HttpDns to solve the problem of playback failure caused by domain hijacking during playback.

Android&iOS: V2TXLivePlayer now supports WebRTC playback.

#### **Function optimizations:**

Android: Added "speed" field to TXVodDownloadMediaInfo for obtaining network download speed in video downloading.

#### **Bug fixes:**

iOS: Fixed issue where picture-in-picture switching repeatedly caused switching to fail.

iOS: Fixed issue where interface still showed video downloading as incomplete after download was completed.

iOS: Fixed issue where video playback was stuck on iOS 16.

Android&iOS: Fixed issue where playback failed on some devices due to high video frame rate. Android&iOS:

Reduced the time it takes for playback to fail in case of network exceptions.

### Player SDK 10.8 @ 2022.10.27

#### **Function optimizations:**

Android&iOS: Added VOD\_PLAY\_EVT\_LOOP\_ONCE\_COMPLETE event for single loop playback completion.



Android: Optimized startup by reducing the number of calls to NetworkInfo.getExtraInfo to comply with regulations.

#### Bug fixes:

Android&iOS: Fixed issue where private encrypted videos failed to play in certain scenarios.

Android&iOS: Fixed issue where some videos failed to play when transmitted through gzip.

Android&iOS: Fixed issue where progress bar duration did not match actual video duration after playback ended.

iOS: Fixed issue where v2 protocol failed to retrieve video source address for appid&fileid playback.

#### Player SDK 10.7 @ 2022.09.20

#### **Function optimizations:**

Android&iOS: Changed startPlay interface for VOD playback to startVodPlay.

Android&iOS: Changed startPlay interface for live playback to startLivePlay.

iOS: Fixed issue where playback could not be resumed after being in background for a long time.

Android: Fixed issue where some videos failed to play on older Android systems.

### Player SDK 10.6 @ 2022.08.31

#### **Function optimizations:**

Android&iOS: Added sprite map and URL information callback for fileid playback.

Android&iOS: Optimized package size.

#### Bug fixes:

iOS: Fixed issue where offline downloaded private encrypted videos failed to play in certain scenarios.

#### Player SDK 10.5 @ 2022.08.12

**Bug fixes:** 

Android&iOS: Fixed issue where short links without video format suffixes caused playback failure.

### Player SDK 10.4.0 @ 2022.07.21

#### Function optimizations:

Android&iOS: Added support for adaptive playback for HLS live streaming.

#### **Bug fixes:**

Android: Fixed abnormal interval between onNetStatus and progress callbacks.

Android: Fixed null pointer exception caused by failure to call setConfig on the player.

iOS: Fixed issue where replaying caused stuttering in certain scenarios.

### Player SDK 10.3.0 @ 2022.07.06

#### New features:

iOS: Added support for picture-in-picture mode during video playback.

#### Bug fixes:



Android: Fixed issue where continuous playback of video lists in the background using hardware decoding was interrupted.

Android&iOS: Fixed issue where seek completion event was not called back.

### Player SDK 10.2.0 @ 2022.06.23

#### Function optimizations:

Android&iOS: Optimized callback parameters such as cachedBytes and IP address during playback.

#### **Bug fixes:**

Android&iOS: Fixed issue where hardware decoding failed for H265 format videos.

Android&iOS: Fixed issue with playing HLS live streaming.

iOS: Fixed issue with abnormal retrieval of supportedBitrates in certain scenarios.

### Player SDK 10.1.0 @ 2022.05.31

Android&iOS: Optimized video super-resolution effect. Android&iOS: Fixed issue with nested m3u8 refer header sub-stream transmission. iOS: Resolved conflict with third-party SDK ffmpeg. Android&iOS: Optimized player kernel performance.

### Player SDK 9.5.29040 @ 2022.05.13

Android&iOS: Fixed issue where playing mp3 with cover image failed.

### Player SDK 9.5.29036 @ 2022.05.06

Android: Fixed issue where SurfaceView caused black screen due to repeated add and remove.

### Player SDK Android 9.5.29035, iOS 9.5.29036 @ 2022.04.28

Android&iOS: Added video preloading function.

Android&iOS: Added ability to pause player before onPrepared event.

Android&iOS: Added ability to maintain pause state when switching stream under pause state. Android&iOS: Optimized playback performance.

### Player SDK 9.5.29016 @ 2022.03.30

Android&iOS: Added support for fine-grained control of cached traffic, preloading buffer and startup buffer can be controlled separately.

Android&iOS: Added ability to specify preferred resolution before startup and find the most suitable resolution to start playback.

### Player SDK 9.5.29015 @ 2022.03.25

Android: Optimized playback performance.

### Player SDK 9.5.29011 @ 2022.03.10

iOS: Optimized version compatibility issues.

#### Player SDK 9.5.29009 @ 2022.03.03

Android&iOS: Added support for terminal ultra-high definition, can be accessed through plugins.

Android&iOS: Optimized private encrypted video playback.

Android&iOS: Optimized accurate seeking to frames.

Android&iOS: Added support for EXT-X-DISCONTINUITY tag in HLS.

Android&iOS: Optimized player kernel and improved performance.

Android&iOS: Player component provides demo for immersive short video, feed video stream, video preview, video cover and video dynamic watermark functions.

#### Player SDK 9.5 @ 2022.01.11

Android: Fixed issue where switching resolution twice after playing any video in the video list to the end caused replay. Android&iOS: Fixed issue where playback time point was inaccurate when playing back at different time points.

#### Player SDK 9.4 @ 2021.12.09

iOS: Fixed issue where switching HLS stream caused black screen.

iOS: Fixed issue where frequent seeking during playback with VOD player caused noise.

Android: Fixed issue where anti-theft chain sprite failed to retrieve.

Android: Fixed issue where VOD player occasionally reported errors during HLS offline download.

Android&iOS: Fixed issue where accurate seeking with player was inaccurate.

#### Player SDK 9.3 @ 2021.11.04

Android&iOS: Fixed issue where enabling preloading with VOD player and calling startPlay caused abnormal sound. Android&iOS: Fixed issue where hardware decoding with VOD player caused callback resolution to be 0 for HEVC videos.

#### Player SDK 9.2 @ 2021.09.26

Android&iOS: VOD player supports HLS reinforcement encryption playback.

Android: Fixed issue where playing addresses with special characters failed.

Android: Fixed issue where frequent switching between foreground and background caused occasional sound without picture.

#### Player SDK 9.1 @ 2021.09.02

Android: Fixed issue where playback crashed on specific Android 5.x devices.

Android: Optimized live playback to prevent overexposure under specific conditions.

#### Player SDK 9.0 @ 2021.08.06

iOS: Fixed issue where enabling smoothSwitchBitrate caused crash when switching resolutions repeatedly. iOS: Optimized VOD player to prevent playback progress from being abnormal after network recovery.

#### Player SDK 8.9 @ 2021.07.15

Android: Fixed issue where callback event logic was incorrect after VOD player lost network connection.

#### Player SDK 8.8 @ 2021.06.21

iOS: Fixed issue where starting and stopping playback with VOD player multiple times caused memory leaks.

Android: Fixed issue where playing HLS files on Android 11 caused errors.

Android: Fixed issue where default live playback was jerky and other live streams occasionally had accelerated sound and picture.

Android&iOS: Fixed issue where seeking with VOD player for specific videos was slow.

Android&iOS: Fixed issue where VOD player displayed slowly after pausing playback and setting progress with seek.

## **Player Adapter**

#### Player Adapter 1.4.0 @ 2023.04.18

Android&iOS: Added support for decrypting VOD CDN encryption.

#### Player Adapter 1.2.0 @ 2022.03.10

Android&iOS: Added support for playing adaptive bitrate, transcoded, and original videos through FileId.

### Player Adapter Release @ 2021.07.22

First release of iOS & Android player component Adapter.

# Release Notes(Flutter)

Last updated : 2023-07-13 14:38:12

## Player SDK Flutter version 11.3.0 @ 2023.07.07

Removed deprecated live time-shifting interface. Fixed known issues.

## Player SDK for Flutter V11.2.0 @ 2023.06.05

Minimum requirement version for Flutter development environment has been raised. Fixed known issues.

## Player SDK for Flutter V11.1.1 @ 2023.05.08

Removed unnecessary third-party dependencies. Fixed known issues.

## Player SDK for Flutter V11.1.0 @ 2023.04.10

Refactored native bridging layer. Fixed known issues.

## Player SDK for Flutter V11.0.0 @ 2023.02.28

Added offline download demo.

Fixed known issues including sprite map disorder and failure to open picture-in-picture on iOS.

## Player SDK for Flutter V10.9.0 @ 2023.01.03

Fixed the issue of picture-in-picture not supporting playback of encrypted videos on Android.

Fixed the issue of automatic playback after returning to the application interface from the background when live streaming or VOD is paused.

Optimized log output for easier issue tracking.

## Player SDK for Flutter V10.8.0 @ 2022.10.20

Adjusted the picture-in-picture interaction of the player component, supporting returning to the previous page after enabling picture-in-picture.

## Player SDK for Flutter V10.7.0 @ 2022.09.20

Changed the startPlay interface for VOD playback to startVodPlay. Changed the startPlay interface for live streaming playback to startLivePlay. Changed the playWithModel interface for the player component to playWithModelNeedLicence.

## Player SDK for Flutter V1.0.3 @ 2022.07.05

Added picture-in-picture (PIP) function for Android and iOS.

## Player SDK for Flutter V1.0.2 @ 2022.06.24

Added the SuperPlayer component with integrated UI solution. Improved the SDK interfaces for live streaming and VOD. Fixed the issue of playback failure when using fileId and psign.

## Player SDK for Flutter released on 2021.07.16

Release Flutter Player SDK.

# Licenses Adding and Renewing a License

Last updated : 2024-05-15 17:21:08

## Overview

The Player SDK provides video playback capabilities for live streaming and VOD. Mobile and Web platforms are separately authorized and billed. You need to access the corresponding License before you can use the respective features.

The access and usage of License are as follows:

#### 1. Purchasing License

You can refer to the product features and purchase guide to confirm and purchase the License you need. If you want to apply for a trial License for testing, please see the free trial guidelines.

#### 2. Bind License

After purchasing a License, you need to bind the newly purchased License with the application/domain you want to authorize to implement authorization for the corresponding application/domain. Mobile platforms are authorized by application package name, and Web platforms by domain name. The guidance on mobile platform binding operations and Web platform binding operations are below.

#### 3. Configure License

After binding, you will obtain the authorization credential **License URL** and **License Key** in the console. During the SDK integration process, you need to enter the corresponding information. Please keep it safe. For the specific method of entry, refer to the integration documentation for each platform.

#### Note:

MLVB License and UGSV License include all the capabilities of the Player Mobile Basic Version License. Therefore, they can also be used to unlock the basic features of the Player Mobile SDK. For more information on MLVB License, see Adding and Renewing a License and Adding and Renewing a License. For more information on UGSV License, see License Pricing Overview and Adding and Renewing a License.

## Add and Renew a License (Mobile)

#### **Purchase an official License**

Before binding the License, you can access the Player License in the following manner:

| Types of License | Acquisition Method | Price (USD) | Validity Period |
|------------------|--------------------|-------------|-----------------|
|                  |                    |             |                 |



| Player Mobile Basic Version<br>License   | Apply for free | 0   | 1 year  |
|--|----------------|-----|---------|
| Player Mobile Premium<br>Version License | Purchase now   | 499 | 1 month |

#### **Bind an official License**

After purchasing, you need to go to the VOD console or CSS console to bind the License to the application to activate it. You can choose to Method 1: create a new official application and select the player License or Method 2: unlock the player feature in an existing application and bind the License as two ways to officially bind the License.

Method 1

Method 2

1. Go to the VOD console or CSS console > License Management > Mobile License, and click Create official license.



2. Enter an app name, a package name, and a bundle ID, select **Player License**, choose **Basic Version** or **Premium Version**, and click **Next**.

| Create officia                               | l license ×   |
|--|---|
| Select o                                     | apabilities for your license > (2) Bind license resources   |
| Basic informa                                | ition   |
| App na<br>or Bun                             | rme, Package name, and Bundle ID are required. You cannot modify the Package name<br>dle ID bound to an official license. |
| App name                                     |   |
|  | Max 128 bytes; supports letters, Chinese characters, numbers, spaces, underscores, hyphens, and periods. E.g.: TRTC       |
| Package name                                 |   |
|  | Max 128 bytes; supports letters, numbers, spaces, underscores, hyphens, and periods. E.g.: tencent.trtc.com               |
| Bundle ID                                    |   |
|  | Max 128 bytes; supports letters, numbers, spaces, underscores, hyphens, and periods.<br>E.g.: tencent.trtc.com            |
| Select capab                                 | lities  |
| MLVB Licer<br>Publish live st                | ISE<br>reams using RTMP, play videos live (from CDNs) or on demand  |
| UGSV Lice                                    | 150   |
| Lite: Shoo                                   | t and edit videos, play videos live (from CDNs) or on demand  |
| <ul> <li>Standard:</li> <li>Lite)</li> </ul> | Filters, special effects, transition effects, and more (in addition to the capabilities of UGSV                           |
|  |   |
| Player Lice                                  | nse   |
|  | streaming and VOD video playback  |
| Basic: live                                  | VD video and more (in addition to the completities of Discon Regio)   |
| Basic: live Premium:                         | VR video and more (in addition to the capabilities of Player Basic)   |
| Basic: live Premium:                         | VR video and more (in addition to the capabilities of Player Basic)   |

3. Go to the **Choose License resources and bind** page, click **Bind now**, select the **Unbound** player License (if there are no bindable License resources, refer to Purchase Player License), and click **Create** to create the application and generate an official License.

| u have selected 1 | capabilities. Please bind license resources for them.       |                                 |
|-------------------|---|---------------------------------|
| License type      | Resource name/ID  | Validity period                 |
| Player Premium    | You haven't bound a license n                               | esourde yet <mark>Bind</mark>   |
|                   | Search by license resource name (such as "live stream publi | shing") or ID (such as "18162") |
|                   | Resource name/ID  | Validity period ↓               |
|                   | Player Premium<br>Resource ID: luvc128da8818e505be88b       | 2024-03-18 to 2024-04-18        |
|                   |   |                                 |
|                   |   |                                 |

4. After the official License is successfully created, the page will display the generated official License information. When initializing SDK configuration, you need to input License URL and License Key. Please save the following information properly. Refer to Configuring and Reviewing License guide to input your License URL and License Key in the SDK to complete License authorization.

| ▼ testtt Official license<br>Package name testtt | Bundle ID testtt Creation time 2024-03    | -25 11:35:36    |                            | Edit |
|--|---|-----------------|----------------------------|------|
| Basic informat<br>License URL<br>License key     |   |                 |                            |      |
| Player Premiur                                   | n   | Update validity |                            |      |
| Status   | Normal                                    |                 |                            |      |
| Feature  | Video Playback Premium                    |                 | Activate more capabilities |      |
| Validity period                                  | 2024-03-18 14:59:44 - 2024-05-19 00:00:00 |                 |                            |      |
| Auto-renewal                                     | Disabled Enable                           |                 |                            |      |
|  |   |                 |                            |      |

- 1. Go to the VOD console or CSS console> License Management > Mobile License.
- 2. Select the Official license that you want to add the player feature, and click Activate more capabilities.



| ackage name 1412 | Bundle ID 14123 Creation time 2024-03-18     | 15:51:06        |                            |
|------------------|--|-----------------|----------------------------|
| Basic informat   | ion  |                 |                            |
| License URL      | ACCOUNTS AND A                               |                 |                            |
| License key      |  |                 |                            |
|                  |  |                 |                            |
| UGSV Standard    | d  | Update validity |                            |
| Status           | Normal                                       |                 |                            |
| Feature          | Video production (standard) + Video playback |                 | Activate more capabilities |
| Validity period  | 2023-04-23 16:50:20 - 2024-05-10 00:00:00    |                 |                            |
|                  | Enabled Disable Manage                       |                 |                            |

3. Select Player License, then click Next.

| Activ | ate more capabilities  | × |
|-------|--|---|
| 1     | Select capabilities for your license > (2) Bind license resources  |   |
| eleo  | t capabilities   |   |
| м     | VB License   |   |
| Pu    | lish live streams using RTMP, play videos live (from CDNs) or on demand  |   |
|       | Standard: Filters, special effects, transition effects, and more (in addition to the capabilities of UGSV<br>Lite) |   |
|       |  | _ |
| Pla   | yer License<br>Basic: live streaming and VOD video plavback  |   |
|       | Premium: VR video and more (in addition to the capabilities of Plaver Basic)                                       |   |
| 0     | riomani. Writhadd and more (in addition to the dapabilition of hayor basis)  |   |
| 0     |  |   |
| 0     |  |   |

4. Enter the **select and bind License** page, click **Bind now**, select the **unbound** player License (if there are no bindable License resources, refer to purchasing a player License), and click **Confirm** to generate the official version of the player feature under the application.

| Select capa         | bilities for your license > 2 Bind license > 2 Bind license bind license resources for them. | cense resources                      |
|---------------------|--|--------------------------------------|
| License type        | Resource name/ID   | Validity period                      |
| Player Premium      | You haven't bound a licen  | ise resource yet <mark>Bind</mark>   |
|                     | Search by license resource name (such as *live stream  | publishing") or ID (such as "18162") |
|                     | Resource name/ID   | Validity period ↓                    |
|                     | Player Premium<br>Resource ID: luvc128da8818e505be88b  | 2024-03-18 to 2024-04-18             |
|                     |  |                                      |
| u can go to the Lic | ense Purchase Page to buy new licenses   |                                      |

#### Note:

Before clicking **Confirm**, double-check the bundle ID and package name and make sure they are identical to what you submit to app stores. **The information cannot be modified after submission**.

#### Update official License valid period

You can log in to the VOD console or the CSS console > License Management > Mobile License page to view the valid period of the Player Mobile official license. You can also subscribe to the SDK in Message Subscription, set up Message Center/Email/SMS and other message receiving channels to receive official license expires reminders. The player official license will send you an expires reminder 32 days, 7 days, 3 days, and 1 day before the current time, reminding you to renew in a timely manner to avoid affecting the business operation.

If your Player Mobile official license has expired, you can apply to renew the validity period **within one month before expiration**. Please refer to the following operations to renew:

#### Update Basic Version License valid period

The Mobile **Basic Version** is free to use. Choose the license you need to update the valid period. If the remaining valid period is within 30 days, you can click renew to extend the valid period for free.

#### Note:

The Player Mobile Basic Version License does not support enabling auto-renewal.



| Package name 1111 | 111 Bundle ID 111111 Creation time 2024-03-2  | 2 16:11:042                           |                            |  |
|-------------------|---|---------------------------------------|----------------------------|--|
| Basic informa     | tion  |                                       |                            |  |
| License URL       |   | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |                            |  |
| License key       | 100 C 100 |                                       |                            |  |
|                   |   |                                       |                            |  |
| Player Basic      |   | Renew                                 |                            |  |
| Status            | Expires in 22 day(s), 11 hour(s), and 54 minute(s)  |                                       | Activate more capabilities |  |
| Feature           | Video Playback  |                                       |                            |  |
| Mellelly needed   | 2024-03-22 18:11:42 - 2024-04-18:00:00:00   |                                       |                            |  |

#### Update Premium Version License valid period

1. Select the target license and click **Update validity** in the **Player** features.



2. You can choose two methods to update the license valid: **Renew the current license** or **Select another license resource to replace**. Specific as follows.

#### Note:

**Resources with auto-renewal enabled do not support the license resource replacement method for renewal.** If you want to change the validity period to that of another license, disable the auto-renewal function.

| Current License<br>acquisition method  | Acquired through direct purcha   | The resources with automatic renewal enabled do<br>not support the 'Select another license resource to<br>'eplace' option for renewal. If you want to change<br>the validity period to another license, please<br>disable the current automatic renewal status.   |  |
|--|--|---|--|
| Renew the current lice   | ise  | Select another license resource to replace  |  |
| The validity period after n<br>period + renewal duratio  | renewal is: current validity<br>n.   | Validity period after replacement: change to the<br>validity period of other License (The original License<br>can be rebound to other applications after being<br>replaced)   |  |
| If selecting "Renew the cur<br>even days, according to Te<br>ne valid period will be refre-<br>nore than seven days, the o<br>and automatically complete | rent license", when the license res<br>incent Cloud's billing regulations, r<br>shed to: current validity period + re<br>priginal resource has been destroyu<br>the replacement binding. After rer | ource is within the valid period or has expired for less than<br>renewing for the current license is supported. After renewal,<br>mewal duration. When the license resource has expired for<br>ed, and we will purchase the same type of resource for you<br>newal, the valid period will be refreshed to: payment date + |  |

Renew the current license

Select another license resource to replace

1. Click Renew the current license, and click Update validity.



| Current License<br>acquisition method   | chase  |
|---|--|
| Renew the current license<br>The validity period after renewal is: current validity<br>period + renewal duration.   | Select another license resource to replace<br>Validity period after replacement: change to the<br>validity period of other License (The original License<br>can be rebound to other applications after being<br>replaced)  |
| If selecting "Renew the current license", when the license re<br>seven days, according to Tencent Cloud's billing regulations,<br>the valid period will be refreshed to: current validity period -<br>more than seven days, the original resource has been destro<br>and automatically complete the replacement binding. After<br>renewal duration.<br>If selecting "Select another license resource to replace", ass<br>replacement is from April 30, 2023, to April 30, 2024, the val<br>After the original license is replaced, if it is still within the val<br>applications. | source is within the valid period or has expired for less than<br>, renewing for the current license is supported. After renewal,<br>+ renewal duration. When the license resource has expired for<br>oyed, and we will purchase the same type of resource for you<br>renewal, the valid period will be refreshed to: payment date +<br>suming that the valid period of the license used for<br>lid period will be refreshed to April 30, 2024 after replacement<br>lid period, it can be used to replace and bind other |

#### Note:

If you select Renew the current license:

When the license resource **is within the valid period or has expired for less than seven days**, according to Tencent Cloud's billing regulations, renewing for the current license is supported. After renewal, the valid period will be refreshed to: **current validity period + renewal duration**.

When the license resource **has expired for more than seven days**, the original resource has been destroyed, and we wil purchase the same type of resource for you and automatically complete the replacement binding, After renewal, the valid period will be refreshed to: **payment date + renewal duration**.

In the Renewal interface, select the Renewal period. The Player Premium Version License renews monthly.
 Click Confirm to extend the license valid period.

| Renewal             |            |                       |                 |            | ×         |
|---------------------|------------|-----------------------|-----------------|------------|-----------|
| License type        | Current Ex | Renewal<br>period (i) | Expiration time | Unit Price | Renewal A |
| Player Premium Lic  | 2024-04-19 | 1 month 🔻             | 2024-05-19      | Querying   | Querying  |
| Total Cost: Queryin | <b>g</b> i |                       |                 |            |           |
|                     |            | Confirm               | Cancel          |            |           |

1. Click Select another license resource to replace, and click Update validity.

| acquisition method   | rchase  |
|--|---|
| Renew the current license The validity period after renewal is: current validity   | Select another license resource to replace<br>Validity period after replacement: change to the  |
| period + renewal duration.   | validity period of other License (The original License<br>can be rebound to other applications after being<br>replaced)   |
| f selecting "Renew the current license", when the license r<br>even days, according to Tencent Cloud's billing regulations<br>he valid period will be refreshed to: current validity period<br>nore than seven days, the original resource has been destr<br>nd automatically complete the replacement binding. After<br>answal duration | esource is within the valid period or has expired for less than<br>s, renewing for the current license is supported. After renewal,<br>+ renewal duration. When the license resource has expired for<br>royed, and we will purchase the same type of resource for you<br>renewal, the valid period will be refreshed to: payment date +<br>suming that the valid period of the license used for |

2. In the **Update validity** interface, click **Bind**. Select the unbound Player Advanced license (if there is no available resource pack to bind, you can go to License Purchase Page to buy), and click **Confirm**.

| Jurrent license i | mormation   |  |
|-------------------|---|--|
| Current license   | Player Premium  |  |
| Expires on        | 2024-04-19  |  |
| Bind license res  | burces  |  |
| License type      | Resource name/ID                                      | Validity period                          |
| Player Premium    | You haven't bound a lic                               | ense resour <mark>e yet Bind</mark>      |
|                   | Search by license resource name (such as "live strea  | m publishing") or ID (such as "18162") Q |
|                   | Resource name/ID                                      | valiaity perioa 🔶                        |
|                   | Player Premium<br>Resource ID: luvc128da8818e505be88b | 2024-03-18 to 2024-04-18                 |
|                   |   |  |
|                   |   |  |

3. Check the renewed validity period.

#### Note:

The official license of the player does not support information modifications. If you need to make modifications to the license information, after purchasing the resource package, please do not use it for the update of the valid license period. Click to **Create official license and bound the license** to recreate the application, add a new license, and bound the new bundle ID information.

#### **Upgrade Basic to Premium Version License**

If you have Activated the Mobile Player **Basic Version License** and require External Subtitles, Advanced Picture-in-Picture Components, and other premium features, you can follow the instructions below to upgrade to the Mobile Player **Premium Version License**, unlocking more features:

1. Select the official Mobile Player **Basic Version License** you want to upgrade and click on the **Upgrade**.

| Package name | 11111 Bundle ID | 11111 Cre       | eation time May | 11, 2024 11:45:55 ( | UTC+08:00) Asia/Shanghai |
|--------------|-----------------|-----------------|-----------------|---------------------|--------------------------|
| Basic infor  | rmation         |                 |                 |                     |                          |
| License URL  |                 |                 | 100 M 100 M     |                     | <u> </u>                 |
| License key  |                 |                 | <u> </u>        |                     |                          |
|              |                 |                 |                 |                     |                          |
| Player Bas   | ic              |                 |                 |                     | Upgrade                  |
| Status       | Normal          |                 |                 |                     |                          |
| Feature      | Video Playbac   | k               |                 |                     |                          |
| Start time   | May 11, 2024    | 11:45:55 (UTC+0 | (00:80)         |                     |                          |
| End time     | May 12, 2025    | 00:00:00 (UTC+0 | 8:00)           |                     |                          |

2. Enter the upgrade interface and click **Bind**. Then select the Player Premium License you want to bind and click **Confirm** to upgrade to the Mobile Player **Basic Version License**.

| Upgrade                           |   | ×                                    |
|-----------------------------------|---|--------------------------------------|
| Upgrade informatio                | n   |                                      |
| Current edition F<br>Upgrade to F | Player Basic<br>Player Premium                            |                                      |
| Bind license resour               | ces   |                                      |
| License type                      | Resource name/ID Validity period                          |                                      |
| Player Premium                    | er Premium You haven't bound a license resource vet Bind  |                                      |
|                                   | Search by license resource name (such as "live stream put | olishing") or ID (such as "18162") Q |
|                                   | Resource name/ID  | Validity period ↓                    |
|                                   | Player Premium<br>Resource ID                             | Mar 14, 2024 to Jul 14, 2024         |
|                                   | Player Premium<br>Resource ID:                            | Mar 18, 2024 to May 18, 2024         |
| You can go to the Licens          | Player Premium<br>Resource ID                             | Mar 18, 2024 to May 18, 2024         |
|                                   | Confirm Cancel  |                                      |

## Add and Renew a License (Web)

## Purchase an official License

Before binding the license, you can refer to the following methods to obtain the player license:

| Types of License                      | Authorization Unit  | Acquisition<br>Method | Price<br>(USD) | Validity<br>Period |
|---------------------------------------|---|-----------------------|----------------|--------------------|
| Player Web Basic Version<br>License   | Precise Domain (1 license can<br>authorize up to 10 precise<br>domains) | Apply for free        | 0              | 1 year             |
| Player Web Premium<br>Version License | Wildcard Domain (1 License can authorize 1 wildcard domain)             | Purchase now          | 99             | 1 month            |

### Bind an official license

After purchase, you need to bind the license to the application in VOD console or CSS console > License

Management > Web License to activate it.

1. Enter the Web License, and click on Create Official License.

|                         | Free                 |
|-------------------------|----------------------|
| Create official license | Create trial license |
|                         | •                    |

2. Select the version. The player's Web official license includes Basic and Premium Versions.

The Basic Version can be applied for free, with a valid period of one year. It only supports **precise domain** bound, and can be associated with up to **10**.

The Premium Version is for monthly paid use. It only supports wildcard domains bound, and can be associated with 1.

Add Basic Version License

Add Premium Version License

#### Note:

1 Web Basic Version License can be bound to up to 10 precise domains.

You can add domains at any time before reaching the limit.

Bound domains cannot be modified.

1. Select Player Basic, fill in the Project name and Domain , and click Next.
| 1 Basic information and capabilities       >       2 Choose the resource and bin         Edition       Player Basic         Support basic features including video playback for live streaming & VOD video, HLS private encryption and so on. You can apply for free with one year validity period and bind precise domain name.       In addition to basic features, it also supports VR video, security check and so on. You can use it after purchasing Player Premium and it supports to bind wildcard domain name.         Project Name       Please input your project name, eg: Player | × |
|---|---|
| Edition Player Basic Support basic features including video playback for live streaming & VOD video, HLS private encryption and so on. You can apply for free with one year validity period and bind precise domain name. Project Name Please input your project name, eg: Player   | d |
| Support basic features including video playback for live streaming & VOD video, HLS private encryption and so on. You can apply for free with one year validity period and bind precise domain name.       In addition to basic features, it also supports VR video, security check and so on. You can use it after purchasing Player Premium and it supports to bind wildcard domain name.         Project Name       Please input your project name, eg: Player   |   |
| Project Name Please input your project name, eg: Player   |   |
|   |   |
| Support English, Chinese, number, space, _, -, ., up to a maximum of 128 bytes  |   |
| Domain Please input your domain name, eg: tencent.com   |   |
| Only precise domain name bound is supported, up to a maximum of 128 bytes   |   |
| + Add   |   |
| One Web basic license can bind a maximum of ten precise domain names.<br>Support to add a new domain name before reaching the limit.<br>The bound domain name cannot be edited.   |   |
| Next  |   |

2. Automatically select the **1-year free Web Player Basic Version License** resource, and click create to create the application and generate the license.

| Basic information and capabilities > 2 Choose the resource and bind |  |                 |  |
|---|--|-----------------|--|
| u have selected 1   | capabilities. Please bind license resources for then | ı.              |  |
| License type  | Resource name/ID                                     | Validity period |  |
| Player Basic  | Web Player Basic License [Free]                      | One year        |  |
|   |  |                 |  |
|   |  |                 |  |
|   |  |                 |  |
|   |  |                 |  |
|   |  |                 |  |

3. After the official license is successfully created, the page will display the generated official license information. When initializing the SDK configuration, you need to input the License URL and License Key parameters, please properly save the following information.



| Authorization Information  | Player Basic   | Rene |
|----------------------------|--|------|
| License URL<br>License key | Status         Normal           Feature         Web Video Playback           Validity period         2024-03-22 12:58:03 - 2025-03-23 00:00:00 |      |
|                            |  |      |

#### Note :

1 Web Premium Version License can be associated with 1 wildcard domain and the domain cannot be modified once bound.

1. Select Player Premium, fill in the Project name and Domain , and click Next.

| 1 Basic i    | information and capabilities $>$  | 2 Choose the resource and bind  |  |
|--------------|---|---|--|
| Edition      | Player Basic  | Player Advanced   |  |
|              | Support basic features including<br>video playback for live streaming &<br>VOD video, HLS private encryption<br>and so on. You can apply for free with<br>one year validity period and bind<br>precise domain name. | In addition to basic features, it also<br>supports VR video, security check<br>and so on. You can use it after<br>purchasing Player Premium and it<br>supports to bind wildcard domain<br>name. |  |
| Project name | test  | un la a maximum of 100 kulas  |  |
| Domain       | *.test.com  | -, ., up to a maximum or r26 bytes  |  |
|              | Only wildcard domain name bound is supported, up to a maximum of 128 bytes  |   |  |
|              | One Web premium license can bind one wilde<br>edited after bound.   | card domain name, and the name cannot be  |  |

2. Enter **the selection of license resources and binding page**, click **Bind**, select the unbound player license (if there is no bindable license resource, you can refer to Purchase Player License), and click **Create** to create the application and generate the Advanced license.

| Create Player official Basic informati You have selected 1 capa | l license<br>ion and capabilities > 2<br>bilities. Please bind license resources for the | Choose the resource and bind                    |
|---|--|---|
| License type F  | Resource name/ID   | Validity period                                 |
| Player Premium  | You haven't bour   | nd a license resource yet <b>Bind</b>           |
| _   | Search by license resource name (such as "liv  | e stream publishing") or ID (such as "18162") Q |
|   | Resource name/ID   | Validity period 🗼                               |
|   | Player Premium<br>Resource ID: luvcc83dd18e3c6badd                                       | b 2024-03-14 to 2024-06-14                      |
|   | Player Premium     Resource ID: luvc44efdd518e505d06                                     | 48 <b>2024-03-18 to 2024-05-18</b>              |
| You can go to the Licens  | Player Premium<br>Resource ID: luvc1329f0718e505d06                                      | 48 <b>2024-03-18 to 2024-04-18</b>              |
|   | Player Premium<br>Resource ID: luvc2419e4a18e505d0i                                      | 348 <b>2024-03-18 to 2024-04-18</b>             |
| 03  | Player Premium<br>Resource ID: luvc47b750718e505d0                                       | 647 2024-03-18 to 2024-04-18                    |

Upon the successful creation of the Official License, the page will display the produced Official License information.
 This information, including the License URL and License Key, is required when configuring the SDK initialization.
 Please make sure to securely store this information.



#### Update official License valid period

You can log in to **VOD console** or **CSS console** > **License Management** > **Mobile License** page to view the valid period of the Player Web official license. You can also subscribe to the SDK in Message Subscription, set up **Message Center/Email/SMS** and other message receiving channels to receive official license expires reminders. The player official license will send you an expires reminder 32 days, 7 days, 3 days, and 1 day before the current time, reminding you to renew in a timely manner to avoid affecting the normal business operation. If your Player Web official license has expired, you can apply to renew the validity period within one month before expiration. Please refer to the following operations to renew:

#### Update Basic Version License valid period

The Web **Basic Version** is free to use. Choose the license you need to update the valid period. If the remaining valid period is within 30 days, you can click **Renew** to extend the valid period for free.

#### Note:



The Player Web Basic Version License does not support enabling auto-renewal.

|                 |   | Can be renewed when the left validity period <= 30 days |
|-----------------|---|---|
| Player Basic    |   | Renew   |
| Status          | Normal                                    |   |
| Feature         | Web Video Playback                        |   |
| Validity period | 2024-03-22 12:58:03 - 2025-03-23 00:00:00 |   |

#### Update Premium Version License valid period

1. Select the target license and click **Update validity** in the **Player** features.

| Authorization Information | Player Prem  | nium   | Undate validit |
|---------------------------|--|--|----------------|
| License URL License Key   | Status<br>Feature<br>Validity period<br>Auto-reneval | Normal<br>Web Video Playback Premium<br>2024-03-14 18:04:33 - 2024-05-15 00:00:00<br>Disabled Enable |                |
| Domain Information        |  |  |                |

2. The Player Premium Version License supports 2 methods of updating the valid period: **Renew the current license** or **Select another license resource to replace**. The specific methods are as follows.

#### Note:

**Resources with auto-renewal enabled do not support the license resource replacement method for renewal**. If you want to change the validity period to that of another license, disable the auto-renewal function.

| Current License Acquired through direct purch<br>acquisition method  | The resources with automatic renewal enabled do<br>not support the 'Select another license resource to<br>replace' option for renewal. If you want to change<br>the validity period to another license, please<br>disable the current automatic renewal status.   |
|--|---|
| Renew the current license<br>The validity period after renewal is: current validity<br>period + renewal duration.  | Select another license resource to replace<br>Validity period after replacement: change to the<br>validity period of other License (The original License<br>can be rebound to other applications after being<br>replaced)   |
| If selecting "Renew the current license", when the license reven days, according to Tencent Cloud's billing regulations he valid period will be refreshed to: current validity period + nore than seven days, the original resource has been destroind automatically complete the replacement binding. After renewal duration.<br>If selecting "Select another license resource to replace", as eplacement is from April 30, 2023, to April 30, 2024, the vale eplacement. After the original license is replaced, if it is still ther applications.<br>Both methods have no difference in license functionality. An ethanged. For more detail, please refer to UGSV SDK Docum | esource is within the valid period or has expired for less thar<br>i, renewing for the current license is supported. After renewal<br>renewal duration. When the license resource has expired for<br>yed, and we will purchase the same type of resource for your<br>renewal, the valid period will be refreshed to: payment date a<br>usuming that the valid period of the license used for<br>lid period will be refreshed to April 30, 2024 after<br>within the valid period, it can be used to replace and bind<br>after updating, the original LicenseUrl and Key will not be<br>mentation 🔀 |

Renew the current license

Select another license resource to replace

1. Click Renew the current license, and click Update validity.

| Stencent Cloud |
|----------------|
|----------------|

| Current License<br>Acquired through direct pure<br>acquisition method   | chase   |
|---|---|
| Renew the current license<br>The validity period after renewal is: current validity<br>period + renewal duration.   | Select another license resource to replace<br>Validity period after replacement: change to the<br>validity period of other License (The original License<br>can be rebound to other applications after being<br>replaced)   |
| If selecting "Renew the current license", when the license re<br>even days, according to Tencent Cloud's billing regulations,<br>he valid period will be refreshed to: current validity period -<br>nore than seven days, the original resource has been destro<br>and automatically complete the replacement binding. After<br>enewal duration.<br>If selecting "Select another license resource to replace", ass<br>eplacement is from April 30, 2023, to April 30, 2024, the val<br>After the original license is replaced, if it is still within the val<br>applications. | source is within the valid period or has expired for less than<br>renewing for the current license is supported. After renewal,<br>renewal duration. When the license resource has expired for<br>yed, and we will purchase the same type of resource for you<br>renewal, the valid period will be refreshed to: payment date +<br>uming that the valid period of the license used for<br>id period will be refreshed to April 30, 2024 after replacement<br>lid period, it can be used to replace and bind other |

#### Note:

If you select Renew the current license:

When the license resource is within the valid period or has expired for less than seven days, according to Tencent Cloud's billing regulations, renewing for the current license is supported. After renewal, the valid period will be refreshed to: current validity period + renewal duration.

When the license resource **has expired for more than seven days**, the original resource has been destroyed, and we wil purchase the same type of resource for you and automatically complete the replacement binding, After renewal, the valid period will be refreshed to: **payment date + renewal duration**.

In the Renewal interface, select the Renewal period. The Player Premium Version License renews monthly.
 Click Confirm to extend the license valid period.

| Renewal            |             |                       |                 |            | ×         |
|--------------------|-------------|-----------------------|-----------------|------------|-----------|
| License type       | Current Ex. | Renewal<br>period (j) | Expiration time | Unit Price | Renewal A |
| Player Premium Lic | 2024-05-15  | 1 month 🔻             | 2024-06-15      | Querying   | Querying  |
| otal Cost: Queryin | <b>g</b> i  |                       |                 |            |           |
|                    |             | Confirm               | Cancel          |            |           |

1. Click Select another license resource to replace, and click Update validity.

| Current License<br>Acquired through direct pure<br>acquisition method   | chase  |
|---|--|
| Renew the current license<br>The validity period after renewal is: current validity<br>period + renewal duration.   | Select another license resource to replace<br>Validity period after replacement: change to the<br>validity period of other License (The original License<br>can be rebound to other applications after being<br>replaced)  |
| If selecting "Renew the current license", when the license re<br>even days, according to Tencent Cloud's billing regulations,<br>he valid period will be refreshed to: current validity period -<br>nore than seven days, the original resource has been destro<br>ind automatically complete the replacement binding. After<br>enewal duration.<br>If selecting "Select another license resource to replace", ass<br>eplacement is from April 30, 2023, to April 30, 2024, the val<br>After the original license is replaced, if it is still within the val<br>upplications. | source is within the valid period or has expired for less than<br>, renewing for the current license is supported. After renewal,<br>+ renewal duration. When the license resource has expired for<br>yyed, and we will purchase the same type of resource for you<br>renewal, the valid period will be refreshed to: payment date +<br>suming that the valid period of the license used for<br>lid period will be refreshed to April 30, 2024 after replacement<br>lid period, it can be used to replace and bind other |

2. In the **Update validity** interface, click **Bind**. Select the unbound Player Advanced license (if there is no available resource pack to bind, you can go to License Purchase Page to buy), and click **Confirm**.

| Update validity               |   | ×  |
|-------------------------------|---|--|
| Current license in            | formation   |  |
| Current license<br>Expires on | Player Premium<br>2024-05-15                              |  |
| Bind license resou            | urces   |  |
| License type                  | Resource name/ID  | Validity period                          |
| Player Premium                | You haven't bound a license                               | e resourc <mark>e</mark> yet <b>Bind</b> |
|                               | Search by license resource name (such as "live stream put | blishing") or ID (such as "18162") Q     |
|                               | Resource name/ID  | Validity period ↓                        |
|                               | Player Premium<br>Resource ID: luvcc83dd18e3c6baddb       | 2024-03-14 to 2024-06-14                 |
|                               | Player Premium<br>Resource ID: luvc44efdd518e505d0648     | 2024-03-18 to 2024-05-18                 |
| You can go to the Lice        | Player Premium<br>Resource ID: luvc1329f0718e505d0648     | 2024-03-18 to 2024-04-18                 |
| 8:59                          | Player Premium<br>Resource ID: luvc2419e4a18e505d0648     | 2024-03-18 to 2024-04-18                 |
|                               | Player Premium<br>Resource ID: luvc47b750718e505d0647     | 2024-03-18 to 2024-04-18                 |

3. Check the renewed validity period.

#### Note:

The official license of the player does not support information modifications. If you need to make modifications to the license information, after purchasing the resource package, please do not use it for the update of the valid license period. Click to **Create official license and bound the license** to recreate the application, add a new license, and bound the new bundle ID information.

## Auto-renewal

You can manage auto-renewal through **Console** and **Billing center** in 2 methods. The details are as follows.

Console

Billing center

License supports the enabling of automatic renewal. License resources with automatic renewal enabled will be automatically renewed monthly 3 days before expiration. Make sure your account has sufficient available balance before enabling automatic renewal. Otherwise, it may lead to a renewal failure and affect your usage. Log in to the CSS or VOD console > License Management, locate the license you wish to manage for automatic renewal:

1. Enable Auto-renewal.

1.1 In the **Disabled** status of the license Auto-renewal, click to **Enable** auto-renewal, and it will be automatically deducted and renewed **monthly** three days before expiration.



| Package name testtt | Bundle ID testtt Creation         | time 2024-03-25 11:35:36                             |        |
|---------------------|-----------------------------------|--|--------|
| Basic informati     | on                                |  |        |
| License URL         | https://license-test.vod2.vod-qcl | oud.com/test/license/testV2/1300224785_1/v_cube.lice | ense 🗇 |
| License key         | 88120816aff0f3a526803354a5af5     | 5213 🖻   |        |
|                     |                                   |  |        |
| Player Premiun      |                                   | Lindate validity                                     |        |
| Flayer Freihlun     | 1                                 | opdate valuity                                       |        |
| Status              | Normal                            |  |        |
| Feature             | Video Playback Premium            |  |        |
| Validity period     | 05-100-10-11-0001-05-1            | 19 00:00:00  |        |
| Auto-renewal        | Disabled Enable                   |  |        |

| fter you enable auto renewa  | I, the Player Premium li | icense will be automatically | renewed every mon |
|------------------------------|--------------------------|------------------------------|-------------------|
| ree days before it et pires. |                          |                              |                   |
| License type                 | End time                 | Expiration time              | Price             |
| Player Premium License       | 2024-05-19               | 2024-06-19                   | Querying          |

1.2 Auto-renewal status changed to Enabled.

| Player Premiur  | n   | Update validity |
|-----------------|---|-----------------|
| Status          | Normal                                    |                 |
| Feature         | Video Playback Premium                    |                 |
| Validity period | 2024_03_18_14:59:44 - 2024-05-19 00:00:00 |                 |
| Auto-renewal    | Enabled Disable Manage                    |                 |

2. Disable Auto-renewal. The **Auto-renewal** of the license can be turned off in the **Enabled** status by click **Disable**. After it expires, it will no longer be automatically renewed.



| Player Premiun  | n   | Update validi |
|---|---|---------------|
| Status  | Normal  |               |
| Feature   | Video Playback Premium  |               |
| Validity period   | 2024-03-18 14:59:44 - 2024-05-19 00:00:00                           |               |
| Auto-renewal  | Enabed Disable Nanage   |               |
|   |   |               |
|   |   |               |
|   | ×   |               |
| sable<br>er you disable auto-renewal, th  | A he Player Premium license will no longer be automatically renewed |               |
| sable<br>er you disable auto-renewal, then it expires.<br>ou want to continue to use the                        | A Player Premium license will no longer be automatically renewed    |               |
| sable<br>sr you disable auto-renewal, th<br>en it expires.<br>ou want to continue to use the<br>ore it expires. | A he Player Premium license will no longer be automatically renewed |               |

You can navigate to Renewal Management to set resources to automatic renewal.

In the search box on the right, search for a **Player**, locate the target resource, and click **Set to Auto-Renewal**.

| Manual Renewal (46) Auto-renewal (10) Non-renewal (0) |  |                   |                 | Enter ins  | stance ID or name to searc Q Ø ±    |
|---|--|-------------------|-----------------|------------|-------------------------------------|
| Batch Renewal Set to Auto-Renewal Set to Non-Renewal  |  |                   |                 |            |                                     |
| Instance ID/Name Product Name                         | Region                                 | Expiration Date 1 | Project ▼       | Unit Price | Operation                           |
|   | Other (others)<br>Regardless of Region | 2024-04-14        | DEFAULT PROJECT | -          | Ren w Set to Auto-Renewal<br>More ▼ |



## Configuring and Viewing a License

Last updated : 2024-04-18 17:06:53

## Video Playback License

## Configuration

Before you call the APIs of the Player SDK, follow the steps below to configure the license:

iOS

Add the code below in [AppDelegate application:didFinishLaunchingWithOptions:] :





```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
NSString * const licenceURL = @"<The license URL obtained>";
NSString * const licenceKey = @"<The key obtained>";
//TXLiveBase can be found in the "TXLiveBase.h" header file
[TXLiveBase setLicence:licenceURL key:licenceKey];
[TXLiveBase setObserver:self];
NSLog(@"SDK Version = %@", [TXLiveBase getSDKVersionStr]);
return YES;
}
```



```
#pragma mark - TXLiveBaseDelegate
- (void)onLicenceLoaded:(int)result Reason:(NSString *)reason {
    NSLog(@"onLicenceLoaded: result:%d reason:%@", result, reason);
    // If result is not 0, it means the setting failed and needs to be retried.
    if (result != 0) {
        [TXLiveBase setLicence:licenceURL key:licenceKey];
    }
}@end
```

#### Android

Add the code below in  $% \left( {{\left[ {{{\rm{Add}}\;{\rm{the}}\;{\rm{code}\;{\rm{below}\;in}}} \right]}} \right)$  application :



```
public class MApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        String licenceURL = ""; // The license URL obtained
        String licenceKey = ""; // The license key obtained
        TXLiveBase.getInstance().setLicence(this, licenceURL, licenceKey);
        TXLiveBase.setListener(new TXLiveBaseListener() {
            @Override
            public void onLicenceLoaded(int result, String reason) {
        }
    }
}
```



#### Note :

1. License is a strong online verification logic. When calling TXLiveBase#setLicence after the application is started for the first time, the network must be ensured. When the App is launched for the first time, the Internet access permission may not have been authorized. You need to wait for the Internet access permission to be granted before calling TXLiveBase#setLicence again.

Monitor the TXLiveBase#setLicence loading result: onLicenceLoaded interface. If it fails, retry and guide accordingly according to the actual situation. If it fails multiple times, the frequency can be limited, and the service can be supplemented with product pop-up windows and other guidance to allow users to check the network situation.
 TXLiveBase#setLicence can be called multiple times. It is recommended to call TXLiveBase#setLicence when entering the main interface of the App to ensure successful loading.

4. For multi-process apps, ensure that TXLiveBase#setLicence is called when each process using the player starts. For example: For an Android app that uses an independent process to play videos, if the process is killed by the system and restarted during background playback, TXLiveBase#setLicence must also be called.

#### **Viewing license information**

After the license is successfully configured, you can call the API below to view the license information. Please note that it may take a while for the configuration to take effect. The exact time needed depends on your network conditions. **iOS:** 



NSLog(@"%@", [TXLiveBase getLicenceInfo]);

Android:



TXLiveBase.getInstance().getLicenceInfo();

# Player Guide Stage 1. Play back a source video

Last updated : 2023-06-19 16:19:01

## Overview

This document describes how to upload a video to VOD and play it using VOD's player.

## Prerequisites

Before you start, do the following:

### Activating VOD

Follow the steps below to activate VOD:

- 1. Sign up for a Tencent Cloud account.
- 2. Purchase VOD services. For details, see Billing Overview.
- 3. Go to the **VOD console**.
- At this point, you have activated VOD.

## Step 1. Upload a Video

This step shows you how to upload a video.

1. In the VOD console, select Application Management on the left sidebar and select the target application. On the **Media Assets > Video/Audio Management** page, click **Upload**.

| 🔗 Tencent Cloud                                  | Overview Products - Video on Demand +  |
|--|--|
| VOD  | Video/Audio Management                 |
| Service Overview                                 | Uploaded Uploading                     |
| Media Assets                                     | Upload Batch Delete                    |
| <ul> <li>Video / Audio<br/>Management</li> </ul> | Separate key words with a vertical lin |
| • Image<br>Management                            | Audio/Video name/ID                    |
| Cold Storage                                     | D:2437.                                |
| <ul> <li>Smart bitrate<br/>reduction</li> </ul>  |  |
| Task Management                                  | 00:01:31                               |
| Video Moderation                                 |  |
| System Settings                                  | ID:38 <sup>-</sup>                     |
| Media Processing                                 | Total items: 3                         |
| Playback   |  |

2. On the upload page, select **Local Upload**, and click **Select File** to upload a local video. Set the other fields as follows:

Select No processing after upload for Media processing.

| 🔗 Tencent Cloud                                  | Overview Products - Video on Demand + |                 |
|--|---------------------------------------|-----------------|
| VOD  | Video/Audio Management                |                 |
| Service Overview                                 |                                       |                 |
| ▶ Media Assets ^                                 |                                       | Upload Method 🕻 |
| <ul> <li>Video / Audio<br/>Management</li> </ul> |                                       | Upload          |
| <ul> <li>Image<br/>Management</li> </ul>         |                                       |                 |
| Cold Storage                                     |                                       |                 |
| <ul> <li>Smart bitrate<br/>reduction</li> </ul>  |                                       |                 |
| 🔄 Task Management                                |                                       |                 |
| Video Moderation                                 |                                       |                 |
| System Settings                                  |                                       |                 |
| 🔂 Media Processing 👋                             |                                       |                 |
| Distribution and<br>Playback                     |                                       | Process C       |
| Upload Storage                                   |                                       |                 |
| S Callback Settings                              |                                       | Upload          |

3. Click **Upload**. You will be taken to the **Uploading** page. When **Status** changes to **Uploaded successfully**, the upload is completed, and you can view the **file ID** of the file (for example, 387xxxx8142975036).

| Stencent Cloud                                   | Overview Prod | lucts 👻 Video on Demano | + |                |           |                  |            |  |
|--|---------------|-------------------------|---|----------------|-----------|------------------|------------|--|
| VOD  | Video/Audio   | Management              |   |                |           |                  |            |  |
| Service Overview                                 | Uploaded      | Uploading(1/1)          |   |                |           |                  |            |  |
| ▶ Media Assets ^                                 |               |                         |   | FileId         | File Name | Audio/Video name | Video Size |  |
| <ul> <li>Video / Audio<br/>Management</li> </ul> |               |                         |   | 24             | inp4      | mp4              | 1.49MB     |  |
| <ul> <li>Image<br/>Management</li> </ul>         |               |                         |   | Total items: 1 |           |                  |            |  |

## Step 2. Generate a Player Signature

In this step, you use the signature tool to quickly generate a player signature to play the video.

1. Select **Distribution and Playback > Player Signature Tools** on the left sidebar and complete the following settings:

Video file ID: Enter the file ID generated in Step 1 (387xxxx8142975036).

**Signature expiration time**: Enter the player signature expiration time. If you leave it empty, the signature will never expire.

Playable video type: Select Original.

2. Click **Generate** to get the signature string.

| So Tencent Cloud  | Overview Products + Video on Demand +       |  |
|---|---|--|
| VOD   | Player Signature Tools                      |  |
| Service Overview  | Signature generation Signature verification |  |
| 🕑 Media Assets 🗸  |   | A player signature authorizes a device to play videos using VOD's player. For details, see <u>Player Signature</u> . |
| Task Management   |   |  |
| Video Moderation  |   | Basic settings   |
| System Settings   |   | App ID   |
| 🐼 Media Processing 👋  |   | Video file ID • Edit How do I view the file ID?  |
| Distribution and<br>Playback                                      |   | Current Unix timestamp * 2022-11-24 17:02:54 -> 1669280574 (Unix)  |
| Domain Name   |   | Signature expiration time No expiration if left empty  |
| <ul> <li>Default</li> <li>Distribution</li> <li>Domain</li> </ul> |   | Playback key Manage  |
| Purge and   |   | Playable content   |
| Prefetch  |   | Playback domain 🚯 od2.myqcloud.com 🔻   |
| <ul> <li>Player Signature<br/>Tools</li> </ul>                    |   | Distribution protocol () HTTP •  |
| Upload Storage  |   | Playable video type Original   |
| 😌 Callback Settings   |   | Image sprite ()<br>Screenshot template Dimensions  |
| Data Center   |   |  |
| 🕚 Usage Statistics  |   | SpriteScreenshot (10) 142 x 80   |
| 🚱 Data Analysis   |   |  |
| Download Log  |   | Hotlink protection & encryption  |
| Common tools  |   | Generate   |
| <ul> <li>License</li> <li>Management</li> </ul>                   |   | Signature ey/hbGc/OiJUz1NiIsInR5cCl6lkpXVCJ9.ev/hr+HRI2Cl6bnVsbCwiZmIsZUlkijoiMjQzNzkxNTc2Miov/MTMvMjq5liwiY3VycmV   |
| Activate  |   |  |

## Step 3. Play the Video

After step 2, you have obtained the three parameters needed for video playback: appId , fileId and psign (player signature). The following describes how to play the video on the web.

#### Playback on the web

1. Open the web player demo.

Select Video playback.

Click the File ID tab.

fileID: Enter the same file ID (387xxxx8142975036) in the previous step.

**appID**: Enter the ID of the VOD application to which the file belongs (which is also the App ID displayed on the signature generation page in the previous step).

psign: Enter the signature string generated in the previous step.

2. Click **Preview** to play the video.

| Video playback Thu     | mbnail previews | - auto generate | Thumbnail p      | reviews - upload | Subtitles | Callback              | s Dynamic      | watermark | Vid      |
|------------------------|-----------------|-----------------|------------------|------------------|-----------|-----------------------|----------------|-----------|----------|
| Key hotlink protection | Adaptive bitra  | ate streaming   | Video quality cł | ange messages    | Checkpo   | int restart           | Video playlist | Playbac   | c switch |
| Custom error messages  | Statistics      | Player size     | Custom UI        | Playback speed   | change    | Multi-langua          | ge Multi-in    | stance    | More     |
| 3                      |                 |                 |                  |                  |           | URL                   | 2<br>ile ID    |           |          |
|                        |                 |                 |                  |                  |           | fileID:<br>3877023051 | 3              |           |          |
|                        |                 |                 |                  |                  |           | appID:                | 4              |           |          |
|                        |                 |                 |                  |                  | 0         | psign:                |                |           |          |
|                        |                 |                 |                  |                  |           | eyJhbGciOiJIU         | JzI1NilsI      |           |          |
|                        |                 |                 |                  |                  |           |                       |                |           |          |
|                        |                 |                 |                  |                  |           | Preview               | Reset          |           |          |
|                        |                 |                 |                  |                  |           |                       |                |           |          |

#### Multi-platform player demos

After generating the player signature, you can use our player demos for web, Android, and iOS to play the video. For details, see the source code for the demos.

## Summary

At this point, you have understood how to upload a video to VOD and play it back in the player. See also:

Stage 2. Play back a transcoded video.

Stage 3. Play back an adaptive bitrate streaming video.

Stage 4. Play back an encrypted video

Stage 5. Play back a long video

# Stage 2. Play back a transcoded video

Last updated : 2023-05-15 17:35:16

## Overview

This document describes how to transcode a video and use the player to play the transcoded video. Before you go on, make sure you have read Stage 1. Play back a source video. You will be using the account registered and the video uploaded in stage 1.

## Step 1. Encrypt a Video

1. In the VOD console, select **Application Management** on the left sidebar and select the target application. On the **Media Assets > Video/Audio Management** page, select the target video (file ID 387xxxx8142975036), and click **Transcoding**.

- 2. Complete the following settings:
- 2.1 Select Transcoding for Processing Type.
- 2.2 Click Select template to select a transcoding template.

| <ul> <li>Audio/Video processing will incur fees. For details, see Media Processing Billing.</li> <li>Processing Type</li> <li>Transcoding</li> <li>Moderation</li> <li>Task Flow</li> <li>Transcoding Template</li> <li>Transcoding Template</li> <li>Mo waterma</li> </ul> | ×  | Process   |
|---|--|---|
| Processing Type Adaptive Bitrate Streaming<br>Moderation Task Flow<br>Transcoding Template Transcoding Template Common Template<br>Watermark Template   | es. For details, see <u>Media Processing</u> | <ol> <li>Audio/Video processing will incur<br/><u>Billing</u>.</li> </ol> |
| Transcoding Template Transcoding Template Common Template Watermark Template No waterma   | ) Adaptive Bitrate Streaming<br>) Task Flow  | Processing Type Orranscoding Moderation                                   |
| Watermark Template No waterma 💌   | 2<br>Diate Common Template                   | Transcoding Template Transcoding Template                                 |
|   |  | Watermark Template No waterma 🔻   |
| Thumbnail I Thumbnail   |  | Thumbnail 🔽 Thumbnail   |
| Confirm Cancel  | Cancel                                       | Confirm   |

| elect    | t transcoding template                             |                                       |    |   | Selected (1)   |               |
|----------|--|---------------------------------------|----|---|----------------|---------------|
| Ente     | er a template name/ID                              |                                       | Q, |   | Template Nam   | Container For |
| -        | Template Name/ID                                   | Resolution (px)                       |    |   | STD-H264-MP4-  |               |
|          | STD-H264-MP4-360P<br>100010<br>General Transcoding | Width (auto-scaled) x<br>Height (360) | 1  |   | 540P<br>100020 | mp4           |
| <b>_</b> | STD-H264-MP4-540P<br>100020<br>General Transcoding | Width (auto-scaled) x<br>Height (540) |    | ÷ |                |               |
|          | STD-H264-MP4-720P<br>100030<br>General Transcoding | Width (auto-scaled) x<br>Height (720) |    |   |                |               |
|          | STD-H264-MP4-1080P<br>100040                       | Width (auto-scaled) x                 | +  |   |                |               |

2.4 Click **Confirm** to start the transcoding task.

| Process                           |  | × |
|-----------------------------------|--|---|
| Audio/Video  <br><u>Billing</u> . | processing will incur fees. For details, see <u>Media Processing</u>                                       |   |
| Processing Type                   | <ul> <li>Transcoding</li> <li>Adaptive Bitrate Streaming</li> <li>Moderation</li> <li>Task Flow</li> </ul> |   |
| Transcoding Template              | Transcoding Template Common Template   |   |
| Watermark Template                | No waterma 🔻   |   |
| Thumbnail                         | ✓ Thumbnail  |   |
|                                   | Confirm  |   |

3. Select **Task Center** on the left sidebar and find the transcoding task you started. If the status of the task has changed from "Processing" to "Completed", the video has been transcoded.

| S Tencent Cloud  | Overview Products - | Video on Demand | + |      |                                   |                                  |                       |
|------------------|---------------------|-----------------|---|------|-----------------------------------|----------------------------------|-----------------------|
| VOD              | Task Management     |                 |   |      |                                   |                                  |                       |
| Service Overview |                     |                 |   |      |                                   |                                  |                       |
| ▶ Media Assets ~ |                     |                 |   | ()   | The Task Management page only dis | plays the details of executed ta | sks and only supports |
| Task Management  |                     |                 |   | Task | ID                                | Task Status <b>T</b>             | Creation Time         |
| Video Moderation |                     |                 |   |      | -b31502b3                         | Completed                        | 2022-11-24 17:2       |
|                  |                     |                 |   |      | 203130203                         | - Completed                      | 2022 11 24 112        |

4. Return to the Media Assets > Video/Audio Management page, find your video, and click Manage on the right.



| 🔗 Tencent Cloud                                  | Overview Products * Video on Demand + |   |                           |                     |      |
|--|---------------------------------------|---|---------------------------|---------------------|------|
| VOD  | Video/Audio Management                |   |                           |                     |      |
| Service Overview                                 | Uploaded Uploading                    |   |                           |                     |      |
| ▶ Media Assets                                   |                                       | Upload Batch Delete Process Quick                                   | Edit More 🔻               |                     |      |
| <ul> <li>Video / Audio<br/>Management</li> </ul> |                                       | Separate key words with a vertical line " " and separate filter con | ditions by pressing Enter | Q                   |      |
| <ul> <li>Image<br/>Management</li> </ul>         |                                       | Audio/Video name/ID   | Audio/Video status        | Audio/Video categ T | Uplo |
| Cold Storage                                     |                                       | mp4<br>ID:24  | ⊘ Normal                  | Other               | 2022 |
| Smart bitrate                                    |                                       | 00:00:06  |                           |                     |      |

Under the Transcoding outputs tab, you will see the output of the transcoding task.

| Standard Transcoding List |        |            |    |
|---------------------------|--------|------------|----|
| Delete                    |        |            |    |
| Specification             | Format | Resolution | Bi |
| STD-H264-MP4-540P         | MP4    | 960 x 540  | 71 |

## Step 2. Generate a Player Signature

In this step, you use the signature tool to quickly generate a player signature to play the video.

1. Select **Distribution and Playback** > **Player Signature Tools** on the left sidebar and complete the following settings:

Video file ID: Enter the file ID generated in Step 1 (387xxxx8142975036).

**Signature expiration time**: Enter the player signature expiration time. If you leave it empty, the signature will never expire.

Playable video type: Select Transcoding.

Transcoding template: Select STD-H264-MP4-540P (100020) .

2. Click **Generate** to get the signature string.

| VOD   | Player Signature Tools |                        |                           |   |   |              |
|---|------------------------|------------------------|---------------------------|---|---|--------------|
| Service Overview  | Signature generation   | Signature verification |                           |   |   |              |
| Media Assets  |                        |                        |                           | uthorizer a device to play videos using VOD's play    | r For dataile, cae Diavar Signature         |              |
| Task Management   |                        |                        |                           | actionizes a device to play videos dating vob s plays | n for details, see <u>mayer signature</u> . |              |
| Video Moderation  |                        |                        | Basic settings            |   |   |              |
| system Settings   |                        |                        | App ID                    |   |   |              |
| 🚯 Media Processing 👋  |                        |                        | Video file ID *           | 243 Edit How do I vi                                  | ew the file ID?                             |              |
| Distribution and<br>Playback                                      |                        |                        | Current Unix timestamp *  | 2022-11-24 17:34:00 -> 1669                           | 282440 (Unix)                               |              |
| Domain Name   |                        |                        | Signature expiration time | No expiration if left empty                           |   |              |
| <ul> <li>Default</li> <li>Distribution</li> <li>Domain</li> </ul> |                        |                        | Playback key              | Manage  |   |              |
| <ul> <li>Purge and<br/>Prefetch</li> </ul>                        |                        |                        | Playable content          | /od2.mygcloud.com ▼                                   |   |              |
| <ul> <li>Player Signature<br/>Tools</li> </ul>                    |                        |                        | Distribution protocol ()  | HTTP V  |   |              |
| Upload Storage  |                        |                        | Playable video type       | Transcoding 👻   |   |              |
| S Callback Settings   |                        |                        | Transcoding template      | Template name/ID                                      | Container Format                            |              |
| Data Center   |                        |                        |                           |   | MD4   |              |
| Usage Statistics  |                        |                        |                           | 310-H204-WP4-340P (100020)                            | WP4   |              |
| 😔 Data Analysis   |                        |                        | Image sprite 🚯            | Screenshot template                                   | Dimensions                                  |              |
| Download Log  |                        |                        |                           |   |   |              |
| Common tools  |                        |                        |                           |   | The specifi                                 | ad file ID ( |
| License<br>Management   |                        |                        | Hotlink protection &      | encryption  |   |              |
| Activate<br>Subapplication  |                        |                        | 4<br>Generate             | 6   |   |              |

## Step 3. Play the Video

After step 2, you have obtained the three parameters needed for video playback: appId , fileId and psign (player signature). The following describes how to play the video on the web.

#### Playback on the web

1. Open the web player demo.

Select Video playback.

Click the File ID tab.

fileID: Enter the same file ID in the previous step.

**appID**: Enter the ID of the VOD application to which the file belongs (which is also the App ID displayed on the signature generation page in the previous step).

psign: Enter the signature string generated in the previous step.

2. Click **Preview** to play the video.

| urun es | Video playback Thu     | mbnail previews - auto generat | te Thumbnail p   | reviews - upload | Subtitles | Callba     | acks Dynami    | c waterma |
|---------|------------------------|--------------------------------|------------------|------------------|-----------|------------|----------------|-----------|
|         | Key hotlink protection | Adaptive bitrate streaming     | Video quality cl | hange messages   | Checkpoir | nt restart | Video playlist | Play      |
|         | Custom error messages  | Statistics Player size         | e Custom UI      | Playback speed   | change    | Multi-lang | uage Multi-i   | nstance   |
|         |                        |                                |                  |                  |           |            |                |           |
|         |                        |                                |                  |                  |           |            | _              |           |
|         |                        |                                |                  |                  |           | URL        | File ID        |           |
|         |                        |                                |                  |                  | f         | fileID:    |                |           |
|         |                        |                                |                  |                  |           | 387702305  |                |           |
|         |                        |                                |                  |                  | a a       | appID:     | 4              |           |
|         |                        |                                |                  |                  |           | 1500       |                |           |
|         |                        |                                |                  |                  |           | osign:     |                |           |
|         |                        |                                |                  |                  |           | eyJhbGciOi | iJIUzI1NilsI   |           |
|         |                        |                                |                  |                  |           |            |                |           |
|         |                        |                                |                  |                  |           |            |                |           |
|         |                        |                                |                  |                  |           |            |                |           |
|         |                        |                                |                  |                  |           | Preview    | Reset          |           |
|         |                        |                                |                  |                  | I         | Preview    | Reset          |           |
|         |                        |                                |                  |                  | I         | Preview    | Reset          |           |

#### Multi-platform player demos

After generating the player signature, you can use our player demos for web, Android, and iOS to play the video. For details, see the source code for the demos.

## Summary

At this point, you have understood how to transcode a video and use the player to play it.

# Stage 3. Play back an adaptive bitrate streaming video

Last updated : 2023-05-15 17:35:41

## Overview

This document describes how to play an adaptive bitrate streaming video. Specifically, the following will be performed: Streams of different resolutions will be generated. The lowest will be 480p and the highest will be 1080p. A screenshot will be taken from the video and will be used as the thumbnail.

Multiple screenshots will be taken at an interval of 20% and will be used as thumbnail previews.

Before you go on, make sure you have read Stage 1. Play back a source video. You will be using the account registered and the video uploaded in stage 1.

## Step 1. Create an Adaptive Bitrate Streaming Template

1. Log in to the VOD console. Select Application Management on the left sidebar. Select the target application and go to **Media Processing > Template Settings**. Under the **Adaptive Bitrate streaming** tab, click **Create Template**.



| VOD  | Template Settings |              |                   |                     |                        | 0                           |                 |            |        |
|--|-------------------|--------------|-------------------|---------------------|------------------------|-----------------------------|-----------------|------------|--------|
| Service Overview                               | Video Transcoding | TSC Template | Audio Transcoding | Remux               | Adaptive Bitrate Strea | aming Image Processing Temp | ate Watermark   | Screenshot | Ani    |
| Media Assets ~                                 |                   |              |                   | Create Template     | 3                      |                             |                 |            |        |
| 🔄 Task Management                              |                   |              |                   |                     |                        |                             |                 |            |        |
| Video Moderation                               |                   |              | Te                | mplate name/ID      | Muxing Type            | F Encryption Type ▼         | Substream Count | Switch     | from l |
| System Settings                                |                   |              | Ad<br>10          | pative-HLS          | HLS                    | Not encrypted               | 6 substream(s)  | Forbid     |        |
| Media Processing     Template Settings         |                   |              | Ad                | lpative-HLS-FairPla | y HLS                  | FairPlay                    | 6 substream(s)  | Forbid     |        |
| Task Flow Settings                             |                   |              | Ad                | lpative-HLS-Encryp  | t u c                  | SimpleAFS                   | 6 cubetroom(c)  | Forbid     |        |
| <ul> <li>DRM</li> <li>Configuration</li> </ul> |                   |              | 12                |                     | TIES                   |                             | o substream(s)  | Torbid     |        |
| Distribution and<br>Playback                   |                   |              | Ad<br>20          | lpative-DASH        | MPEG-DASH              | Not encrypted               | 6 substream(s)  | Forbid     |        |
| Upload Storage                                 |                   |              | Ad<br>13          | lpative-HLS-Widev   | ine HLS                | Widevine                    | 6 substream(s)  | Forbid     |        |
| S Callback Settings                            |                   |              | SD                | MC-Adpative-HLS     | -Fair HLS              | FairPlay                    | 6 substream(s)  | Forbid     |        |
| Data Center                                    |                   |              | 31                |                     |                        |                             |                 |            |        |
| Data Analysis                                  |                   |              | SD<br>41          | MC-Adpative-DAS     | H-W MPEG-DASH          | Widevine                    | 6 substream(s)  | Allow      |        |
| 📩 Download Log 🛛 🗸                             |                   |              | sec<br>14         | gmentTest<br>28617  | HLS                    | Not encrypted               | 1 substream(s)  | Forbid     |        |
| Common tools                                   |                   |              | Tot               | tal 8 items         |                        |                             |                 |            |        |

2. Click **Add Stream** to add a stream 2 and stream 3 and complete the following settings:

#### Basic Info:

Template name: Type MyTestTemplate .

Muxing format: Select "HLS".

Encryption type: Select "Not encrypted".

Switch from Low Resolution to High Resolution: Disable

| Basic Info                    | 0                       |                                  |                           |
|-------------------------------|-------------------------|----------------------------------|---------------------------|
| Template name *               | MyTestTemplate          |                                  | ${\boldsymbol{ \oslash}}$ |
|                               | It should be a combina  | tion of letters, numbers, spaces | s, underscores (_), hy    |
| Template Description          | Please describe temp    | late                             |                           |
|                               | Up to 15 chars for desc | ription                          |                           |
| Muxing Format                 | HLS                     | ~                                |                           |
| Segment Type                  | ts                      | *                                |                           |
| Encryption Type               | 3<br>Not encrypted      | •                                |                           |
| Switch from Low Resolution to | 4                       |                                  |                           |
| High Resolution               |                         |                                  |                           |

#### Streams

| Stream<br>No. | Video<br>Bitrate | Resolution                            | Frame<br>Rate | Audio Bitrate | Sound<br>Channels |
|---------------|------------------|---------------------------------------|---------------|---------------|-------------------|
| Stream 1      | 512 Kbps         | Long side: 0 px, short side: 480 px.  | 24 fps        | 48 Kbps       | Dual              |
| Stream 2      | 512 Kbps         | Long side: 0 px, short side: 720 px.  | 24 fps        | 48 Kbps       | Dual              |
| Stream 3      | 1,024<br>Kbps    | Long side: 0 px, short side: 1080 px. | 24 fps        | 48 Kbps       | Dual              |

| Substream1   |  |   |            |                   |    |                               |
|--|--|---|------------|-------------------|----|-------------------------------|
| Video parameters   |  |   |            |                   |    |                               |
| Encoding standard *  | H.264  | •   |            |                   |    |                               |
| Video bitrate  | Video bitrate  | Kbps  | 5          |                   |    |                               |
|  | Video bitrate should be empt   | y or be   | tween      | 128 and 35000     |    |                               |
| Resolution 🛈   | Video's long side  | рх  | x          | Videos short side | рх | Set by long and short sides 💌 |
|  | Video's long side should be e  | mpty o  | r betw     | veen 128 and 4096 |    |                               |
| Frame rate   | Video Frame Rate   | fps   |            |                   |    |                               |
|  | Video frame rate is between e  | empty a   | and 10     | 0                 |    |                               |
|  |  |   |            |                   |    |                               |
| Audio Paramotore   |  |   |            |                   |    |                               |
| Audio Parameters   |  |   |            |                   |    |                               |
| Audio Parameters   | AAC  | •   |            |                   |    |                               |
| Audio Parameters   | AAC<br>32000 Hz  | ▼<br>▼  |            |                   |    |                               |
| Audio Parameters<br>Encoding standard *<br>Sample Rate *<br>Audio bitrate                    | AAC<br>32000 Hz<br>Audio bitrate   | ▼<br>▼<br>Kbps  | 5          |                   |    |                               |
| Audio Parameters<br>Encoding standard *<br>Sample Rate *<br>Audio bitrate                    | AAC<br>32000 Hz<br>Audio bitrate<br>Audio bitrate should be empt         | <ul> <li>Kbps</li> <li>y or be</li> </ul>                           | ;<br>tween | 26 and 256        |    |                               |
| Audio Parameters<br>Encoding standard *<br>Sample Rate *<br>Audio bitrate<br>Sound Channel * | AAC 32000 Hz Audio bitrate Audio bitrate should be empt Mono Channel O D | <ul> <li>Kbps</li> <li>Kbps</li> <li>automatic structure</li> </ul> | ;<br>tween | 26 and 256        |    |                               |

3. Click **Create**. An adaptive streaming template that converts a video into three streams is created. The ID of the template is 1430219 .

| Template Name/ID          | Muxing Type 🔻 | Substream Count | Switch from Low Re <b>T</b> | Template Type T |
|---------------------------|---------------|-----------------|-----------------------------|-----------------|
| Adpative-HLS<br>10        | HLS           | 6 substream(s)  | Forbid                      | Preset          |
| Adpative-HLS-Encrypt      | HLS           | б substream(s)  | Forbid                      | Preset          |
| Adpative-MPEG_DASH        | MPEG-DASH     | 6 substream(s)  | Allow                       | Preset          |
| MyTestTemplate<br>1145464 | HLS           | 3 substream(s)  | Forbid                      | Custom          |

## Step 2. Create an Image Sprite Template

1. Go to **Media Processing > Template Settings**. Select the **Screenshot** tab and click **Create Screenshot Template**.

| VOD                                    | Template Settings |              |                  |                   |                                 |         |
|--|-------------------|--------------|------------------|-------------------|---------------------------------|---------|
| Service Overview                       | Video Transcoding | TSC Template | Audio Transcodin | ng Remux          | Adaptive Bitrate Streaming Imag | e Proce |
| ▶ Media Assets ~                       |                   |              |                  | Create Screensh   | 3<br>ot Template                |         |
| 🖸 Task Management                      |                   |              |                  | Template Name/I   | D Screenshot Type               | r       |
| Video Moderation                       |                   |              |                  | SpriteScreenshot  | Image prote server              | chat    |
| System Settings                        |                   |              |                  | 10                | image sprite screens            | snot    |
| 🗞 Media Processing                     |                   |              |                  | SampledScreensho  | ot Campling crossphe            |         |
| Template Settings                      |                   |              |                  | 10                | sampling screensho              |         |
| <ul> <li>Task Flow Settings</li> </ul> |                   |              |                  | TimepointScreensl | not Point-in-time screen        | nshot   |

2. Complete the following settings:

Template name: Type MyTestTemplate .

Screenshot type: Select "Image sprite screenshot".

Small image dimension: 726 px x 240 px.

Sampling interval: 20%.

Rows: 10.

Columns: 10.



3. Click Create. An image sprite template with the ID 131864 is created.



Custom

| MyTestTe | mplate |
|----------|--------|
| 113272   |        |

## Step 3. Create and Start a Task Flow

Now you have an adaptive bitrate streaming template (ID 1430219) and an image sprite template (ID 131864). You also need to create a task flow.

726 x 240

1. Go to Media Processing > Task Flow Settings and click Create Task Flow.

Image sprite screenshot

Task flow name: Enter MyTestProcedure .

Configuration item: Select "Adaptive bitrate streaming", "Screenshot", and "Thumbnail generation".

In the Adaptive bitrate streaming task configuration area, click Add Template, and select the

MyTestTemplate template (ID 1430219) created in step 1.

In the Screenshot task configuration area, click Add Template. Select Image sprite for Method for Taking Screenshot and then select the MyTestTemplate template (ID 131864) created in step 2. In the Configuration of task of capturing cover screenshot area, click Add Template. For Screenshot

Template/ID, select "TimepointScreenshot". For Select by time points, select "Percent", and enter "50".
|  | It should be a comb   | ination of letter   | rs, numbers, hyphens (-), and ur   | nderscores (_) with a length up to 2(  | D chars  |  |
|--|---|---|--|--|--|--|
| Task flow Description  | Please describe ta  | ask flow  |  |  |  |  |
|  | Up to 15 chars for d  | lescription   |  | 0  |  |  |
| Configuration Item   | General Transco   | oding TS  | C Transcoding Remux  | Adaptive Bitrate Streaming   | Screenshot Task                                      | 🗸 Capture C  |
|  | Select at least one c                                       | onfiguration ite  | em   |  |  |  |
| Adaptive bitrate st  | reaming task con  | figuration  |  |  |  |  |
| You can go to "Templa  | te Settings - Adaptive                                      | Bitrate Streamin  | ng Template to view adaptive b   | trate streaming template. Template   | creation is not supporte                             | d for the mome                                     |
| Adaptive Bitr  | ate Streaming Temp  | late/ID   | Muxing Type  | Substream C  | ount   | Switch   |
| ▶ MyTestTemp   | olate(1430219)  | r   | HLS  | 3 substream(s  | ;)   | Forbid   |
|  |   |   |  |  |  |  |
| Add Template   |   |   |  |  |  |  |
| Screenshot Task Co<br>You can go to "Templa  | onfiguration<br>te Settings - Screensho                     | ot" to create a s   | screenshot template and go to '  | Template Settings -> Watermark   | <" to create a watermark '                           | template. After                                    |
| Screenshot Task Co<br>You can go to "Templa<br>Method for T  | onfiguration<br>te Settings - Screensho<br>aking Screenshot | ot" to create a s<br>Screenshot/II  | screenshot template and go to '  | Template Settings -> Watermark<br>Image Format   | <" to create a watermark<br>Image Dir                | template. After o<br>nension                       |
| Screenshot Task Co<br>You can go to "Templa<br>Method for T  | aking Screenshot  | ot" to create a s<br>Screenshot/II<br>MyTestTemp                                      | screenshot template and go to '<br>D<br>plate(131864)  | Template Settings -> Watermark<br>Image Format<br>jpg  | " to create a watermark<br>Image Din<br>726 x 240    | template. After o                                  |
| Screenshot Task Co<br>You can go to "Templa"<br>Method for T<br>Image Sprit<br>Add Template  | onfiguration<br>te Settings - Screensho<br>aking Screenshot | ot" to create a s<br>Screenshot/II<br>MyTestTemp                                      | screenshot template and go to '<br>D<br>plate(131864)  | Template Settings -> Watermark<br>Image Format<br>jpg  | c" to create a watermark<br>Image Din<br>726 x 240   | template. After                                    |
| Screenshot Task Co<br>You can go to "Templa"<br>Method for T<br>Image Sprit<br>Add Template  | onfiguration<br>te Settings - Screensho<br>aking Screenshot | ot" to create a s<br>Screenshot/II<br>MyTestTemp                                      | screenshot template and go to '<br>D<br>plate(131864)  | Template Settings -> Watermark<br>Image Format<br>jpg  | <" to create a watermark<br>Image Dir<br>726 x 240   | template. After o                                  |
| Screenshot Task Co<br>You can go to "Templa"<br>Method for T<br>Image Sprit<br>Add Template  | aking Screenshot  | ot" to create a s<br>Screenshot/II<br>MyTestTemp                                      | screenshot template and go to '<br>D<br>plate(131864)  | Template Settings -> Watermark<br>Image Format<br>jpg  | <" to create a watermark t<br>Image Din<br>726 x 240 | template. After o                                  |
| Screenshot Task Co<br>You can go to "Templat<br>Method for T<br>Image Sprit<br>Add Template  | aking Screenshot  | ot" to create a s<br>Screenshot/II<br>MyTestTemp                                      | screenshot template and go to '<br>D<br>plate(131864)<br>shot<br>screenshot template. After creat                        | Template Settings -> Watermark<br>Image Format<br>jpg  | <" to create a watermark<br>Image Din<br>726 x 240   | template. After o                                  |
| Screenshot Task Co<br>You can go to "Templa"<br>Method for T<br>Image Sprit<br>Add Template<br>Configuration of t<br>You can go to "Templa<br>Screenshot Templa              | ask of capturing c<br>te Settings - Screenshot              | ot" to create a s<br>Screenshot/II<br>MyTestTemp                                      | screenshot template and go to '<br>D<br>plate(131864)<br>•<br>shot<br>screenshot template. After creat                   | Template Settings -> Watermark<br>Image Format<br>jpg<br>ion, click Purge,<br>Image Size                   | <" to create a watermark<br>Image Dir<br>726 x 240   | template. After of nension                         |
| Screenshot Task Co<br>You can go to "Templa"<br>Method for T<br>Image Sprit<br>Add Template  | aking Screenshot  | ot" to create a s<br>Screenshot/II<br>MyTestTemp<br>over screens<br>ot" to create a s | screenshot template and go to '<br>D<br>plate(131864)<br>shot<br>screenshot template. After creat<br>Image Format<br>jpg | Template Settings -> Watermark<br>Image Format<br>jpg<br>ion, click Purge,<br>Image Size<br>Same as sourc  | <" to create a watermark t<br>Image Din<br>726 x 240 | template. After<br>nension<br>Select to<br>Percen  |
| Screenshot Task Co<br>You can go to "Templat<br>Method for T<br>Image Sprit<br>Add Template<br>Configuration of t<br>You can go to "Templat<br>Screenshot Template           | aking Screenshot  | ot" to create a s<br>Screenshot/II<br>MyTestTemp                                      | screenshot template and go to '<br>D<br>plate(131864)<br>screenshot template. After creat<br>Image Format<br>jpg         | Template Settings -> Watermark<br>Image Format<br>jpg<br>ion, click Purge,<br>Image Size<br>Same as source | <" to create a watermark t<br>Image Din<br>726 x 240 | template. After<br>nension<br>Select I<br>Perce    |
| Screenshot Task Co<br>You can go to "Templat<br>Method for T<br>Image Sprit<br>Add Template  | aking Screenshot  | ot" to create a s<br>Screenshot/II<br>MyTestTemp<br>over screens<br>ot" to create a s | screenshot template and go to '<br>D<br>plate(131864)  | Template Settings -> Watermark<br>Image Format<br>jpg<br>ion, click Purge.<br>Image Size<br>Same as source | to create a watermark to mage Din<br>726 x 240       | template. After<br>nension<br>Select t<br>Perce    |
| Screenshot Task Co<br>You can go to "Templa"<br>Method for T<br>Image Sprit<br>Add Template<br>Configuration of t<br>You can go to "Templa"<br>Screenshot To<br>Add Template | aking Screenshot  | ot" to create a s<br>Screenshot/II<br>MyTestTemp<br>over screens<br>ot" to create a s | screenshot template and go to ' D plate(131864) shot screenshot template. After creat Image Format jpg                   | Template Settings -> Watermark<br>Image Format<br>jpg<br>ion, click Purge,<br>Image Size<br>Same as source | <" to create a watermark 1<br>Image Din<br>726 x 240 | template. After (<br>nension<br>Select b<br>Percer |

| MyTestProcedure | Custom | 2022-11-24 19:49:34 | 2022-11-24 19:49:34 |
|-----------------|--------|---------------------|---------------------|
|                 |        |                     |                     |

3. Go to **Media Assets > Video/Audio Management**, select the target video (file ID 243xxx814xxxx416), and click **Task Flow**.

| 🔗 Tencent Cloud   | Overview Products  Video on Demand +                                   |
|---|--|
| VOD   | Video/Audio Management<br>2<br>Uploaded Uploading                      |
| Media Assets     Video / Audio     Management           | Upload Batch Delete P<br>Separate key words with a vertical line " " a |
| <ul> <li>Image<br/>Management</li> </ul>                | Audio/Video name/ID  |
| <ul> <li>Cold Storage</li> <li>Smart bitrate</li> </ul> | <sup>3</sup> ↓ 00:00:06 00:00:06                                       |

4. Complete the following settings:

### Select Task Flow as the Processing Type.

Select the "MyTestProcedure" task flow template.

| Process                        |   | ×    |
|--------------------------------|---|------|
| Audio/Vide<br><u>Billing</u> . | o processing will incur fees. For details, see <u>Media Process</u> | sing |
| Processing Type                | Transcoding     Adaptive Ditrate Streaming                          |      |
|                                | O Moderation  |      |
| Task Flow Template             | MyTestProcedure 💌   |      |
|                                | 3<br>Confirm Cancel   |      |

5. Click **Confirm**. Go to **Task Center**. If the status of the task changes from "Processing" to "Completed", the processing of the video is finished.



| OD               | Task Management |         |  |                             |                                       |
|------------------|-----------------|---------|--|-----------------------------|---------------------------------------|
| Service Overview |                 | ~       |  |                             |                                       |
| dia Assets 🗸 🗸   |                 | Û       | The Task Management page only displays the | he details of executed task | s and only supports querying the task |
| gement           |                 | Task ID |  | Task Status 🔻               | Creation Time 🕈                       |
| eration          |                 |         | scedurev2-ac4eeb31502b3 ि                  | Completed                   | 2022-11-24 17:24:21                   |

6. Return to the Media Assets > Video/Audio Management page, find your video, and click Manage on the right.

| 🔗 Tencent Cloud                                  | Overview Prod | lucts + Video on Demand | + |   |  |                   |       |
|--|---------------|-------------------------|---|---|--|-------------------|-------|
| VOD  | Video/Audio   | Management              |   |   |  |                   |       |
| Service Overview                                 | Uploaded      | Uploading               |   |   |  |                   |       |
| ▶ Media Assets                                   |               |                         |   | Upload Batch Delete Process                         | s Quick Edit More 🔻                        |                   |       |
| <ul> <li>Video / Audio<br/>Management</li> </ul> |               |                         |   | Separate key words with a vertical line " " and sep | parate filter conditions by pressing Enter | Q                 |       |
| • Image  |               |                         |   | Audio/Video name/ID                                 | Audio/Video status                         | Audio/Video categ | ♥ Upl |
| Cold Storage                                     |               |                         |   | D24   | ⊘ Normal                                   | Other             | 202   |
| Smart bitrate                                    |               |                         |   | 000006  |  |                   |       |

6.1 In the Basic Info area:

You can view the thumbnail generated and the outputs of adaptive bitrate streaming (template ID: 1430219).



### 6.2 Select the **Screenshots** tab:

You can view the image sprite generated (template ID: 131864).

| mage spine screens |                       |      |         |               |
|--------------------|-----------------------|------|---------|---------------|
| Template ID        | Small Image Dimension | Rows | Columns | Sampling Mode |
| 131864             | 726 x 240             | 10   | 10      | Percent       |

## Step 4. Generate a Player Signature

In this step, you can use the signature tool to quickly generate a signature for the player to play back the video. Select **Distribution and Playback** > **Player Signature Tools** on the left sidebar and complete the following settings:

Video file ID: Enter the file ID (243xxx814xxxx416) in step 3.

**Signature expiration time**: Enter the player signature expiration time. If you leave it empty, the signature will never expire.

Playable video type: Select Unencrypted adaptive bitrate.

Adaptive bitrate template: Select MyTestTemplate (1430219) .

Image sprite: Select MyTestTemplate (131864) .

Click Generate to get the signature string.

## Step 5. Play the Video

After step 4, you have obtained the three parameters needed for video playback: appId , fileId and psign (player signature). The following describes how to play the video on the web.

### Playback on the web

Open the web player demo. Select Video playback. Click the File ID tab. fileID: Enter the same file ID (243xxx814xxxx416) in the previous step. appID: Enter the ID of the VOD application to which the file belongs (which is also the App ID displayed on the signature generation page in the previous step). psign: Enter the signature string generated in the previous step. Click Preview to play the video.

| Key hotlink protection       Adaptive bitrate streaming       Video quality change messages       Checkpoint restart       Video p         Custom error messages       Statistics       Player size       Custom UI       Playback speed change       Multi-language         Image: Custom error messages       Statistics       Player size       Custom UI       Playback speed change       Multi-language         Image: Custom error messages       Statistics       Player size       Custom UI       Playback speed change       Multi-language         Image: Custom error messages       Statistics       Player size       Custom UI       Playback speed change       Multi-language         Image: Custom error messages         Image: Custom error messages       Image: Custom error messages       Image: Custom error messages       Image: Custom error messages       Image: Custom error messages         Image: Custom error messages       Image: Custom error messages       Image: Custom error messages       Image: Custom error messages       Image: Custom error messages         Image: Custom error messages       Image: Custom error messages       Image: Custom error messages       Image: Custom error messages       Image: Custom error messages         Image: Custom error messages | playlist Playbac |            |
|--|------------------|------------|
| Custom error messages       Statistics       Player size       Custom UI       Playback speed change       Multi-language         Image: Custom Error messages       Statistics       Player size       Custom UI       Playback speed change       Multi-language   |                  | ack switch |
| URL File ID   S87702305. aplIC:   150( 150(   psign:   | Multi-instance   | More       |
|  | 3                |            |
| Preview  | Reset            |            |

### Multi-platform player demos

After generating the player signature, you can use our player demos for web, Android, and iOS to play the video. For details, see the source code for the demos.

## Summary

At this point, you have understood how to play an adaptive bitrate streaming video. To learn how to play an encrypted video, see <u>Stage 4</u>. Play back an encrypted video.

# Stage 4. Play back an encrypted video

Last updated : 2023-05-15 17:36:27

## Overview

This document describes how to encrypt a video and use VOD's player to play the encrypted video. Before you go on, make sure you have read Stage 1. Play back a source video. You will be using the account registered and the video uploaded in stage 1.

## Step 1. Encrypt a Video

1. In the VOD console, select Application Management on the left sidebar and select the target application. On the **Media Assets > Video/Audio Management** page, select the target video, and click **Task Flow**.



2. Complete the following settings:

Select Task Flow as the Processing Type.

Select the SimpleAesEncryptPreset task flow template.

| Process                        |   | × |
|--------------------------------|---|---|
| Audio/Vide<br><u>Billing</u> . | o processing will incur fees. For details, see <u>Media Processing</u>                  |   |
| Processing Type                | <ul> <li>Transcoding</li> <li>Adaptive Bitrate Streaming</li> <li>Moderation</li> </ul> |   |
| Task Flow Template             | SimpleAesEncryptPreset 2<br>Confirm Cancel  |   |

### Note:

SimpleAesEncryptPreset is a preset task flow, which uses the adaptive bitrate streaming template 12, thumbnail generation template 10, and image sprite generation template 10.

The adaptive bitrate streaming template 12 outputs encrypted multi-bitrate streams.

3. Click **Confirm**. Go to **Task Center**. If the status of the task changes from "Processing" to "Completed", the processing of the video is finished.

4. Go to Media Assets > Video/Audio Management, find your video, and click Manage on the right.

| 🔗 Tencent Cloud                                  | Overview Prod | ucts <del>v</del> | Video on Demand | StreamLive | StreamPackage | StreamLink   | Cloud Streaming Services         | Tencent Real-Tim        | e Communication +         |                   |        |
|--|---------------|-------------------|-----------------|------------|---------------|--------------|----------------------------------|-------------------------|---------------------------|-------------------|--------|
| VOD  | Video/Audio   | Manage            | ement           |            |               |              |                                  |                         |                           |                   |        |
| Service Overview                                 | Uploaded      | Upload            | ing             |            |               |              |                                  |                         |                           |                   |        |
| Media Assets                                     |               |                   |                 |            |               | Upload       | Batch Delete                     | Process Quic            | k Edit More 🔻             |                   |        |
| <ul> <li>Video / Audio<br/>Management</li> </ul> |               |                   |                 |            |               | Separate key | words with a vertical line " " a | and separate filter cor | ditions by pressing Enter | Q                 |        |
| <ul> <li>Image<br/>Management</li> </ul>         |               |                   |                 |            |               | Audio/\      | 'ideo name/ID                    |                         | Audio/Video status        | Audio/Video categ | T Uplo |
| Cold Storage                                     |               |                   |                 |            |               |              | ID:24                            | -                       | 🕑 Normal                  | Other             | 2022   |
| <ul> <li>Smart bitrate</li> </ul>                |               |                   |                 |            |               | 00:00        | 06                               |                         |                           |                   |        |

### 4.1 In the **Basic Info** area:

You can view the thumbnail generated and the outputs of adaptive bitrate streaming (template ID: 12).

| 🔗 Tencent Cloud                                  | Overview Proc | ducts - Video on I | Demand StreamLive   | StreamPackage | StreamLink    |
|--|---------------|--------------------|---------------------|---------------|---------------|
| VOD  | Video,        | /Audio Managen     | nent                |               |               |
| Service Overview                                 | Basic Info    | Screenshot Info    | Superplayer Preview | Web player co | de generation |
| ▶ Media Assets                                   |               |                    |                     |               | Basic Info    |
| <ul> <li>Video / Audio<br/>Management</li> </ul> |               |                    |                     |               | Thumbnail     |
| <ul> <li>Image<br/>Management</li> </ul>         |               |                    |                     |               |               |
| Cold Storage                                     |               |                    |                     |               | Thumbnail UR  |
| Adaptive Bitrate Streaming L                     | ist           |                    |                     |               |               |
| Add Subtitle Set Bind S                          | Subtitle Set  |                    |                     |               |               |
| Video Info                                       |               |                    |                     |               |               |
| Template name/ID                                 | Size          | Muxing Type        | DRM Type            | Substre       | am Count      |
| MyTestTemplate<br>1430219                        | 1.06MB        | HLS                | None                | 3 substre     | eam(s)        |
| Adpative-HLS-Encrypt<br>12                       | 1.59MB        | HLS                | SimpleAES           | б substre     | eam(s)        |

### 4.2 Select the **Screenshots** tab:

You can view the image sprite generated (template ID: 10).

| mage Sprite Screensho | t List                |      |         |               |
|-----------------------|-----------------------|------|---------|---------------|
| Template ID           | Small Image Dimension | Rows | Columns | Sampling Mode |
| 131864                | 726 x 240             | 10   | 10      | Percent       |
| 10                    | 142 x 80              | 10   | 10      | Time          |

## Step 2. Generate a Player Signature

In this step, you can use the signature tool to quickly generate a signature for the player to play back the video.



1. In the VOD console, select Application Management on the left sidebar and select the target application. Go to **Distribution and Playback > Player Signature Tools** and complete the following settings:

Video file ID: Enter the file ID generated in Step 1.

**Signature expiration time**: Enter the player signature expiration time. If you leave it empty, the signature will never expire.

Playable video type: Select Encrypted adaptive bitrate.

Encryption type: Select \*\*Private (SimpleAES)\*\*.

Adaptive bitrate template: Select Adpative-HLS-Encrypt (12) .

Image sprite: Select SpriteScreenshot (10) .

2. Click **Generate** to get the signature string.

## Step 3. Play the Video

After step 2, you have obtained the three parameters needed for video playback: appId , fileId and psign (player signature). The following describes how to play back the video on the web.

### Playback on the web

Open the web player demo.
 Select Video playback.
 Select the File ID tab.
 fileID: Enter the same file ID in the previous step.
 appID: Enter the ID of the VOD application to which the file belongs (which is also the App ID displayed on the signature generation page in the previous step).

**psign**: Enter the signature string generated in the previous step.

2. Click **Preview** to play the video.

| Key hotlink protection       Adaptive bitrate streaming       Video quality change messages       Checkpoint restart       Video playlist         Custom error messages       Statistics       Player size       Custom UI       Playback speed change       Multi-language       Multi-in  | ires | Video playback Th      | umbnail previews | - auto generate | Thumbnail p      | reviews - upload              | Subtitles | s Callba  | acks    | Dynamic v          | vaterma |
|---|------|------------------------|------------------|-----------------|------------------|-------------------------------|-----------|---|---------|--------------------|---------|
| Custom error messages       Statistics       Player size       Custom UI       Playback speed change       Multi-language       Multi-in         Image: Custom UI       Image: Custom UI       Playback speed change       Image: Custom UI       Image: Cu |      | Key hotlink protection | Adaptive bitra   | ate streaming   | Video quality cl | Video quality change messages |           | int restart   | Video   | eo playlist Playba |         |
|   |      | Custom error messages  | Statistics       | Player size     | Custom UI        | Playback speed                | change    | Multi-lang  | uage    | Multi-ins          | tance   |
| Preview Reset   |      |                        |                  |                 |                  |                               |           |   |         | 6                  |         |
|   |      |                        |                  |                 |                  |                               |           | URL<br>fileID:<br>387702305<br>appID:<br>1500<br>psign:<br>eyJhbGciOi | File ID |                    |         |

### Multi-platform player demos

After generating the player signature, you can use our player demos for web, Android, and iOS to play the video. For details, see the source code for the demos.

## Summary

At this point, you have understood how to encrypt a video and use the player to play it back.

# Stage 5. Play back a long video

Last updated : 2022-12-30 16:30:24

This document describes how to use the player to play back long videos common on audio/video platforms. It covers the web, iOS, and Android versions of the player and also details the features of key hotlink protection, adaptive bitrate streaming, video thumbnail preview, and video timestamping.

## Overview

After reading this document, you will know:

- How to configure key hotlink protection, which allows you to set the validity period, number of viewers, playback duration, etc.
- How to output adaptive bitstreams in VOD (a player can dynamically select the most appropriate bitrate for playback based on the current bandwidth).
- How to set video timestamps.
- How to use an image sprite as a thumbnail in VOD.
- How to use the player.

Before reading this document, make sure that you have read Stage 1. Play back a video with Player in the Player Guide and understand the concept of fileid in VOD.

## Directions

### Step 1. Enable key hotlink protection

The following takes enabling key hotlink protection for the default distribution domain name under your account as an example:

Note :

Do not directly enable hotlink protection for the domain name in your production environment; otherwise, playback of videos in the production environment may fail.

1. Log in to the VOD console, select **Distribution and Playback** > **Domain Name** to enter the settings page.



| 🔗 Tencent Cloud   | Overview Product | s ≠ Video on Dem | and StreamLive | StreamPackage | StreamLink C                 | loud Streaming Services                                  | Tencent Real-Time Commun | nication | +   |  |   |                   |           |             |
|---|------------------|------------------|----------------|---------------|------------------------------|--|--------------------------|----------|---|--|---|-------------------|-----------|-------------|
| VOD   | Domain Mana      | jement           |                |               |                              |  |                          |          |   |  |   |                   |           |             |
| B Service Overview                                      |                  |                  |                |               | Add Domain                   |  |                          |          |   |  |   |                   |           |             |
| 🗈 Media Assets 🗸  |                  |                  |                |               | Domain Name                  | -  |                          | Status   |   | CNAME (1)                                      | Domain Name Type                          |                   | Operation |             |
| Task Management   |                  |                  |                |               | Domaintenic                  |  |                          | Status   |   |  | bonnam runne rype                         |                   | operation |             |
| Video Moderation  |                  |                  |                |               | 8                            | nyqcloud.com   | ault Distribution Domain | ØEnable  |   | 1312895349.vod2.myqcloud.com.cdn.dns<br>v1.com | Preset VOD Domain Nar                     | ne                | Set 2     |             |
| System Settings   |                  |                  |                |               | Total items: 1               |  |                          |          |   |  |   | 10 🔻 / page       |           | /1 page → H |
| 🚯 Media Processing 👋                                    |                  |                  |                |               |                              |  |                          |          |   |  |   |                   |           |             |
| Distribution and Playback                               |                  |                  |                |               | FAQs:                        |  |                          |          |   |  |   |                   |           |             |
| Domain Name   |                  |                  |                |               | Domain Na                    | me Guide   |                          |          | CNAME Guide                               |  | Configuration                             | Suide             |           |             |
| <ul> <li>Default<br/>Distribution<br/>Domain</li> </ul> |                  |                  |                |               | How do I add<br>How do I mod | a custom domain name?<br>fify the default distribution o | domain name?             |          | What is CNAME?<br>What is a CNAME record? |  | How do I finish ICF<br>How do I configure | filing?<br>CNAME? |           |             |

2. Click the **Access Control** tab, find **Key Hotlink Protection**, click the gray button to enable it, click **Generate** in the pop-up window to generate a random key, and click **OK** to save the configuration and make it take effect.

| Enable Key Hotlink     | Protection |        |               | ×   |
|------------------------|------------|--------|---------------|-----|
| Hotlink Protection Key | Confirm    | Cancel | 1<br>Generate | ble |

### Step 2. Output adaptive bitstream and image sprite

This step describes how to transcode a video to adaptive bitstream and output image sprite.

1. Log in to the VOD console, select Video Processing Settings > Template Settings > Adaptive Bitrate Streaming Template, and click Create Template.



| VOD                                       | Template Settings |              |                  |                              |                       |                          |                        |                      |                       |                     |                              |
|---|-------------------|--------------|------------------|------------------------------|-----------------------|--------------------------|------------------------|----------------------|-----------------------|---------------------|------------------------------|
| E Service Overview                        | Video Transcoding | TSC Template | Audio Transcodii | ng Remux Adapt               | ive Bitrate Streaming | Image Processing Templat | e Watermark Scre       | enshot Animated Imag | e Moderation Template | e                   |                              |
| Media Assets ~                            |                   |              |                  | 3                            |                       |                          |                        |                      |                       |                     | Search by template name/ID Q |
| Task Management                           |                   |              | L. L.            | create template              |                       |                          |                        |                      |                       |                     |                              |
| Video Moderation                          |                   |              |                  | Template name/ID             | Muxing Type T         | Encryption Type T        | Substream Count        | Switch from Lo T     | Template Type T       | Creation Time \$    | Operation                    |
| System Settings                           |                   |              |                  | Adpative-HLS<br>10           | HLS                   | Not encrypted            | 6 substream(s)         | Forbid               | Preset                | 2020-02-26 16:36:45 | View Edit Delete             |
| 🛞 Media Processing 👔                      |                   |              |                  |                              |                       |                          |                        |                      |                       |                     |                              |
| Template Settings                         |                   |              |                  | Adpative-HLS-FairPlay        | HLS                   | FairPlay                 | 6 substream(s)         | Forbid               | Preset                | 2020-02-26 16:36:45 | View Edit Delete             |
| Task Flow Settings                        |                   |              |                  | Adpative-HLS-Encrypt         | LU S                  | SimpleAFS                | 5 substream(s)         | Forbid               | Brocot                | 2020-02-26 16:26:45 | Manu Edit Dalata             |
| <ul> <li>DRM<br/>Configuration</li> </ul> |                   |              |                  |                              | 1163                  |                          | o substream(s)         | TODIA                | Preser                | 2020 02 20 10:30:40 | There conc prenete           |
| Distribution and                          |                   |              |                  | Adpative-DASH<br>20          | MPEG-DASH             | Not encrypted            | 6 substream(s)         | Forbid               | Preset                | 2020-02-26 16:36:45 | View Edit Delete             |
| Upload Storage                            |                   |              |                  | Adpative-HLS-Widevine        | HLS                   | Widevine                 | 6 substream(s)         | Forbid               | Preset                | 2022-04-25 17:00:33 | View Edit Delete             |
| S Callback Settings                       |                   |              |                  |                              |                       |                          |                        |                      |                       |                     |                              |
| Data Center                               |                   |              |                  | SDMC+Adpative+HLS+Fair<br>31 | HLS                   | FairPlay                 | 6 substream(s)         | Forbid               | Preset                | 2022-08-25 16:17:52 | View Edit Delete             |
| Usage Statistics                          |                   |              |                  | SDMC-Adpative-DASH-W         | 1050 D401             | Mideologia               | C and a barrier of the | A.B                  | D1                    | 2022 02 25 45 40 22 | Mary offer Delate            |
| 😔 Data Analysis                           |                   |              |                  | 41                           | WIFEG-DASH            | widevine                 | o subsitieani(s)       | Allow                | Preser                | 2022-08-23 10:16:05 | view cuit Devete             |
| 🖬 Download Log 🗸                          |                   |              |                  | segmentTest<br>1428617       | HLS                   | Not encrypted            | 1 substream(s)         | Forbid               | Custom                | 2022-11-02 15:28:49 | View Edit Delete             |
| Common tools                              |                   |              |                  | Total 8 items                |                       |                          |                        |                      |                       | Lines per page:     | 10 - 1/1 > 1                 |

Create the adaptive bitstream through the template as needed. The following example shows an adaptive bitrate streaming template named testAdaptive, which contains three substreams with a resolution of 480p, 720p, and 1080p. The video bitrate, video frame rate, and audio bitrate are the same as the original video.

| Adaptive Bitra    | te Streaming [ | Detail    |                                     |            |                 |               |                         |          | ×         |
|-------------------|----------------|-----------|-------------------------------------|------------|-----------------|---------------|-------------------------|----------|-----------|
| Template name     | testAdaptiv    | ve Mu     | ixing Type HLS                      |            | Encryption Type |               | No encryp               | tion     |           |
| Template ID       | 1430690        | Ten       | nplate Type Custom                  |            | Switch from Low | Resolution to | Forbid                  |          |           |
| Template Descript | tion -         |           |                                     |            | High Resolution |               | <ol> <li>(i)</li> </ol> |          |           |
| Substream         | Video Enc      | Codec Tag | Video Resolution                    | Video bitr | Video Fra       | Audio Enc     | Audio bitr              | Audio Sa | Audio Cha |
| Substream1        | H.264          | -         | scaling<br>proportionally x<br>480  | 0          | Ofps            | AAC           | 0 Kbps                  | 32000 Hz | 2         |
| Substream2        | H.264          | -         | scaling<br>proportionally x<br>720  | 0          | Ofps            | AAC           | 0 Kbps                  | 32000 Hz | 2         |
| Substream3        | H.264          | -         | scaling<br>proportionally x<br>1080 | 0          | Ofps            | AAC           | 0 Kbps                  | 32000 Hz | 2         |
|                   |                |           |                                     | Cano       | cel             |               |                         |          |           |

2. Select Video Processing Settings > Template Settings > Screenshot Template and click Create Template.



| STencent Cloud               | Overview Products * | Video on Demand | d StreamLive StreamPackage | StreamLink Cloud Streaming Se | rvices Tencent Real-Time Communica | tion +                            |                        |                     |                              |
|------------------------------|---------------------|-----------------|----------------------------|-------------------------------|------------------------------------|-----------------------------------|------------------------|---------------------|------------------------------|
| VOD                          | Template Settings   |                 |                            |                               |                                    | 0                                 |                        |                     |                              |
| Service Overview             | Video Transcoding   | TSC Template    | Audio Transcoding Remux    | Adaptive Bitrate Streaming    | Image Processing Template Wa       | termark Screenshot Animated Ima   | ge Moderation Template |                     |                              |
| 🕑 Media Assets               |                     |                 |                            | Create Screenshot Template    |                                    |                                   |                        |                     | Search by template name/ID Q |
| Task Management              |                     |                 |                            | Template Name/ID              | Screenshot Type T                  | Image Size (width*height) (in px) | Template Type 🔻        | Creation Time \$    | Operation                    |
| Video Moderation             |                     |                 |                            | SpriteScreenshot              |                                    | 112 00                            |                        | 2017 01 01 00 00 00 |                              |
| System Settings              |                     |                 |                            | 10                            | image sprite screenshot            | 142 X 80                          | system preset          | 2017-01-01 00:00:00 | view Edit Delete             |
| 🛞 Media Processing           |                     |                 |                            | SampledScreenshot             | Sampling screenshot                | Same as source                    | System preset          | 2017-01-01 00:00:00 | View Edit Delete             |
| Template Settings            |                     |                 |                            |                               |                                    |                                   | -,                     |                     |                              |
| Task Flow Settings           |                     |                 |                            | TimepointScreenshot           | Point-in-time screenshot           | Same as source                    | System preset          | 2017-01-01 00:00:00 | View Edit Delete             |
| • DRM                        |                     |                 |                            |                               |                                    |                                   |                        |                     |                              |
| Configuration                |                     |                 |                            | MyTestTemplate                | Image sprite screenshot            | 726 x 240                         | Custom                 | 2022-11-24 19:44:00 | View Edit Delete             |
| Distribution and<br>Playback |                     |                 |                            | 131864                        |                                    |                                   |                        |                     |                              |

Create the image sprite through the template as needed. The following example shows an image sprite template named testSprite, with a sampling interval of 5%, 10 rows, and 10 columns.

| Template Settings | 😵 vqlmotest 💌 SubA              | .ppld: 1500000417              |                               |   |                           |            |                     | User Guide                        | Ten |
|-------------------|---------------------------------|--------------------------------|-------------------------------|---|---------------------------|------------|---------------------|-----------------------------------|-----|
| Video Transcoding | TESHD Template Audio Transcodir | ig Remux Ad                    | daptive Bitrate Streaming     | Image Processing Temp                     | late Watermark            | Screenshot | Dynamic Image       | Audit Template                    |     |
|                   | Create Screenshot Template      |                                |                               |   |                           |            |                     | Search by template name/ID        | 2   |
|                   | Template Name/ID                |                                |                               |   |                           |            | Creation Time 🕈     | Operation                         |     |
|                   | SpriteScreenshot<br>10          | Template Detai                 | tert <sup>c</sup> orite       | Screenshot Tune                           | Image covite coreenshot   | ×          | 2017-01-01 00:00:00 | View Edit Delete                  |     |
|                   | SampledScreenshot<br>10         | Template ID                    | 116508                        | Image Dimension                           | 142 x scaling proportiona | illy       | 2017-01-01 00:00:00 | <b>View</b> Edit Delete           |     |
|                   | TimepointScreenshot<br>10       | Template Type<br>Creation Time | Custom<br>2021-12-29 18:54:01 | Template Description<br>Sampling Interval | - 5 %                     |            | 2017-01-01 00:00:00 | View Edit Delete                  |     |
|                   | testSprite                      | Modification time              | 2021-12-29 18:54:08           | Rows                                      | 10                        |            | 2021 12 20 10 54 01 | Mary Edit Dalate                  |     |
|                   |                                 |                                |                               | Columns                                   | 10                        |            | 2021-12-29 16:54:01 | View Edit Delete                  |     |
|                   |                                 |                                |                               | 填充方式                                      | 留黒                        |            |                     |                                   |     |
|                   | Total 4 items                   |                                | Edit                          | Cancel                                    |                           |            |                     | Lines per page: 10 • 16 4 1/1 + 1 |     |

3. Add the adaptive bitrate streaming and image sprite templates through the task flow.

Select Video Processing > Task Flow Settings and click Create Task Flow.



| 🔗 Tencent Cloud      | Overview | Products v | Video on Demand | StreamLive | StreamPackage | StreamLink  | Cloud Streaming Services | Tencent Real-Time Communication | + |                     |
|----------------------|----------|------------|-----------------|------------|---------------|-------------|--------------------------|---------------------------------|---|---------------------|
| VOD                  | Task Flo | w Settings |                 |            |               |             |                          |                                 |   |                     |
| Service Overview     |          |            |                 |            |               | Create Tas  | sk Flow                  |                                 |   |                     |
| Media Assets ~       |          |            |                 |            |               | Task Flow M | Name                     | Task Flow Type 🔻                |   | Creation Time 🕈     |
| Task Management      |          |            |                 |            |               | LongVideoP  | Preset                   | Preset                          |   | 2017-01-01 00:00:00 |
| C Video Moderation   |          |            |                 |            |               | SimpleAesE  | ncryptPreset             | Preset                          |   | 2017-01-01 00:00:00 |
| System Settings      |          |            |                 |            |               |             |                          |                                 |   |                     |
| 🚯 Media Processing 🗠 |          |            |                 |            |               | WidevineFai | irPlayPreset             | Preset                          |   | 2022-04-25 17:00:38 |
| Template Settings    |          |            |                 |            |               | SDMC-Wide   | evineFairPlavPreset      | Preset                          |   | 2022-08-25 16:17:32 |
| Task Flow Settings   |          |            |                 |            |               |             | ,                        |                                 |   |                     |
| DRM                  |          |            |                 |            |               | MyTestProce | edure                    | Custom                          |   | 2022-11-24 19:49:34 |

Add a task through the task flow as needed. The following example shows a task flow named testPlayVideo, which only adds the adaptive bitrate streaming and image sprite templates from the previous examples.

| Task Flow Name                             | testPlayVideo<br>It should be a combi                                 | nation of letters, nu                    | umbers, hyphens (-), and | underscores ( ) with a length up to 20 | ) chars                                |                                  |                        |   |
|--|---|--|--------------------------|--|--|----------------------------------|------------------------|---|
| Task flow Description                      | Please describe tas<br>Up to 15 chars for de                          | ik flow<br>escription                    |                          |  |  |                                  |                        |   |
| Configuration Item                         | General Transcoo  | ling TSC Tra                             | anscoding Remux          | Adaptive Bitrate Streaming             | Screenshot Task Capture                | Cover Task Dynamic Image T       | ask 🗌 Video Moderation |   |
| Adaptive bitrate s                         | <b>streaming task conf</b><br>ate Settings - A <mark>daptive f</mark> | <b>iguration</b><br>Bitrate Streaming Te | emplate to view adaptive | bitrate streaming template. Template   | creation is not supported for the mor  | hent.                            |                        | × |
| Adaptive Bit                               | trate Streaming Templa  | ate/ID                                   | Muxing Type              | Substream Co                           | ount Switc                             | h from Low Resolution to High    | Operation              |   |
| ▶<br>testAdaptiv                           | ve(1430690) 💌   | ]  | HLS                      | 3 substream(s                          | ) Forbic                               | I                                | Add Watermark Delete   |   |
| Add Template                               |   |  |                          |  |  |                                  |                        |   |
| Screenshot Task C<br>You can go to "Templa | <b>Configuration</b><br>ate Settings - Screensho                      | t" to create a scree                     | nshot template and go to | ) "Template Settings -> Watermark      | " to create a watermark template. Afte | r creation, click here to Purge. |                        | × |
| Method for                                 | Taking Screenshot   | Screenshot/ID                            |                          | Image Format                           | Image Dimension                        | Time point/Sample int            | erval Operation        |   |
| ▶ Image Sprit                              | te v  | testSprite(13211                         | 8) 🔻                     | jpg                                    | Same as source                         | 5%                               | Delete                 |   |
| Add Template                               |   |  |                          |  |  |                                  |                        |   |
| Submit                                     |   |  |                          |  |  |                                  |                        |   |

4. Select Media Assets > Video/Audio Management, select the target video (FileId: 387xxxx8142975036), click Task Flow, and select a task flow template to start the task.

| Process   | ×   |
|---|-----|
| Audio/Video processing will incur fees. For details, see <u>Media Process</u><br><u>Billing</u> . | ing |
| Processing Type Transcoding Adaptive to ate Streaming Moderation Task Flow                        |     |
| Task Flow Template testPlayVideo  |     |
| Confirm Cancel  |     |

5. At this point, you can view the task execution status in the **Task Managment** page and get the task result after completion.

| 🔗 Tencent Cloud  | Overview Products * | Video on Demand | StreamLive | StreamPackage | StreamLink Cloud Streaming Service | s Tencent      | Real-Time Communicati     | in +                          |  |                     |           |
|------------------|---------------------|-----------------|------------|---------------|------------------------------------|----------------|---------------------------|-------------------------------|--|---------------------|-----------|
| VOD              | Task Management     |                 |            |               |                                    |                |                           |                               |  |                     |           |
| Service Overview |                     |                 |            |               | The Tack Management page on        | v displays the | details of everyted tasks | and only supports quanting th | e tack nonression status and task details within last  | 72 hours            |           |
| ▶ Media Assets ∨ |                     |                 |            |               | The lask management page on        | y uispiays uie | details of executed tasks | and only supports querying th | ie task processing status and task details within last | 12 10015            |           |
| Task Management  |                     |                 |            |               | Task ID                            |                | Task Status <b>T</b>      | Creation Time \$              | Start Time   | Completion Time ‡   | Operation |
| Video Moderation |                     |                 |            |               |                                    | . 10           | Completed                 | 2022-11-30 17:16:12           | 2022-11-30 17:16:12                                    | 2022-11-30 17:16:22 | Details   |
| System Settings  |                     |                 |            |               | 3b18bd1ba5                         | 2a 10          | Completed                 | 2022-11-30 17:41:53           | 2022-11-30 17:41:53                                    | 2022-11-30 17:42:04 | Details   |

### Step 3. Add video timestamps

This step describes how to add a set of video timestamps.

1. Go to VOD Server APIs > Media Asset Management APIs > ModifyMediaInfo and click Debug to enter the TencentCloud API console for debugging.

| 🔗 Tencent Cloud |
|-----------------|
|-----------------|

| STencent Cloud                    | Overview Products + V                   | deo on Demand StreamLive StreamPackage StreamLink Cloud Streaming Ser | vices Tercent Real-Time Communication + 🖸 Ticlet + Billing Center + English + 🚺 +               |  |  |  |  |
|-----------------------------------|---|---|---|--|--|--|--|
|                                   | d APIs Q                                | ModifyMediaInfo<br>ved 2018-07-17                                     | Code Generating Online Call Signature Generation Parameter Description Feedback Data simulation |  |  |  |  |
| Cloud Virtual Machine             | Media Processing ~<br>APIs              | When your did "Sand Reward" on the Waller Cell uses the               |   |  |  |  |  |
| lighthouse                        | Parameter Template V                    | temporary access key of your account will be used to make online      | FileId  |  |  |  |  |
| Auto Scaling                      | APIs                                    | MFA is required for this action. Note that this action may incur      | RequiredVes.  |  |  |  |  |
| Batch Compute                     | No longer V<br>recommended APIs         | fees. Read the billing documentation carefully before the action.     | Type fining   |  |  |  |  |
| Edge Computing Machine            | Task Flow APIs                          | More Options +  | Orangelon   |  |  |  |  |
| Cloud Object Storage              | Other APIs 🗸 🗸                          |   | Urique media file ID.   |  |  |  |  |
| Tencent Kubernetes Engine         | Distribution APIs ~                     | Input Parameters View Only Required Parameters                        |   |  |  |  |  |
| Tencent Container Registry        | Media Management V<br>APIs              | This parameter is not required for this API                           | Subándid  |  |  |  |  |
| Cloud File Storage                | SearchMedia                             | Field (e) (a) (i)   |   |  |  |  |  |
| Cloud HDFS                        | RestoreMedia                            | string  | requirestino.   |  |  |  |  |
| Cloud Block Storage               | ModifyMediaStorageClass                 | SubAppld (Optional) [+] (Q ()   | Type Integer.   |  |  |  |  |
| Cloud Load Balancer               | ModifyMediaInfo                         | integer   | Descriptors   |  |  |  |  |
| Virtual Private Cloud             | FoldsModeSithbuden Name Cylinal ( 🖉 🖗 🕐 |   |   |  |  |  |  |
|                                   | DescribeMediaInfos                      | string  |   |  |  |  |  |
| Direct Connect                    | DeleteMedia                             | Description (Optional)  | Name  |  |  |  |  |
| Content Delivery Network          | Media Upload APIs 🛛 🗸                   | string  | Regired to.   |  |  |  |  |
| Global Application Acceleration F | Event Notification V<br>APIs            | Classid (Optional) (*) 🕲 🛈  | Type: Soring  |  |  |  |  |
| TencentDB for MySQL               | Al-based Sample V                       |   | Description   |  |  |  |  |
| Cloud Native Database TDSQL-C     | Management APIs                         | ExpireTime (Optional) [•] 🕲 🛈   | Media Rename, which can contain up to 64 characters.  |  |  |  |  |
| TencentD8 for SQL Server          | Domain name V<br>management APIs        |   |   |  |  |  |  |
| TencentDB for PostgreSQL          | Region ~                                | CoverData (Optional) (*) 🙆 🛈  | Description   |  |  |  |  |
| Tencent Distributed SQL           | Management APIs                         |   | Distriction   |  |  |  |  |
| TencentD8 for MongoD8             | Categorization APIs                     | AddKeyFrameDescs.N (Optional) [+] 🛞 🛈                                 | Regirestio.   |  |  |  |  |
| Tcaplus Database                  | Task Management V                       | 1 TimeOffset 🕲 🛈  | Type String.  |  |  |  |  |
| TencentD8 for D8brain             | Statistics APIs                         |   | Description:  |  |  |  |  |
| Data Transmission Service         |   | Content @ ()  | Media file description, which can contain up to 128 characters.                                 |  |  |  |  |
| Real-time Communication           |   |   |   |  |  |  |  |
| Video on Demand                   |   | Add   | Classid   |  |  |  |  |
| Maria Brannian Samina             |   | DeleteKeyFrameDescs.N (Optional)                                      | Required Mo.  |  |  |  |  |
| means movessing service           |   | 1 float   | Type: Integer.  |  |  |  |  |
| E Collapse                        |   | 644   | Description:  |  |  |  |  |
|                                   |   |   |   |  |  |  |  |



2. Add the specified video timestamps through the AddKeyFrameDescs.N parameter.



At this point, you have completed the operation in the cloud, output the adaptive bitstreams and image sprite, and added the video timestamps.

### Step 4. Generate a player signature

In this step, you can use the signature tool to quickly generate a signature for the player to play back the video. Select **Distribution and Playback** > **Player Signature Tools** and enter the following information:

- Video file ID: Enter the FileId (387xxxx8142975036) used in step 2.
- **Signature expiration time**: Enter the player signature expiration time. If you leave it empty, the signature will never expire.
- Playable video type: Select Unencrypted adaptive bitrate.
- Playable adaptive bitrate streaming template: Select testAdaptive (1429229) .
- Image sprite template for thumbnail preview: Select testSprite (131353) .
- Hotlink protection and encryption: Toggle it on and configure as follows:
- Link expiration time: Set it to the expiration time of the obtained hotlink protection signature.

· Maximum playback IPs: Set the maximum number of IPs allowed for playback.

Click Generate to get the signature string.

### Step 5. Integrate the player

After step 4, you have obtained the three parameters needed for video playback: appId , fileId and psign (player signature).

This step describes how to play back the adaptive bitstreams, thumbnails, and timestamps in the player for web, iOS, and Android.

- Web
- iOS
- Android

You need to integrate the RT-Cube Player as instructed in Web integration. After importing the player's SDK file, you can play back the video by using the appld, fileId, and psign.

The construction method of the player is TCPlayer, which can be used to create a player instance for playback.

### 1. Place the player container in the HTML file

Place the player container in the desired place on the page. For example, add the following code to index.html (the container ID, width, and height can be customized).

```
<video id="player-container-id" width="414" height="270" preload="auto" playsinli
ne webkit-playsinline>
</video>
```

### 2. Play back with the fileID

Add the following initialization script to the index.html page initialization code to pass in the obtained fileID and appID for playback.

```
var player = TCPlayer('player-container-id', { // player-container-id is the play
er container ID, which must be the same as that in HTML
fileID: '387xxxx8142975036', // `fileID` of the video to be played back
appID: '1400329073', // `appID` of the VOD account to play back the video
psign:'psignxxxx' // `psign` is a player signature. For more information on the s
ignature and how to generate it, see [Player Signature](https://intl.cloud.tencen
t.com/document/product/266/38099).
});
```

## Summary

At this point, you can play back media files with hotlink protection enabled, view the image sprite and video timestamps, and automatically switch dynamic adaptive bitstreams in the player. For more features, see Feature Description.

# Integration (UI Included) Web Integration TCPlayer Integration Guide

Last updated : 2024-06-25 11:48:28

This document introduces the Web Player SDK (TCPlayer) tailored for both VOD and live streaming. It can be quickly integrated with your own web application to enable video playback features. TCPlayer comes with a default set of UI elements, which you can use as needed.

## Overview

The web player utilizes the <video> tag of HTML5 and Flash for video playback. It offers a uniform video playback experience across different platforms when browsers do not natively support video playback. In conjunction with Tencent's Video on Demand service, it provides hotlink protection and features for playing standard encrypted HLS videos.

| Audio/Video<br>Protocol | Use             | URL Format                                  | PC<br>Browser | Mobile<br>Browser   |
|-------------------------|-----------------|---|---------------|---------------------|
| MP3                     | Audio           | http://xxx.vod.myqcloud.com/xxx.mp3         | Supported     | Supported           |
| MP4                     | VOD<br>playback | http://xxx.vod.myqcloud.com/xxx.mp4         | Supported     | Supported           |
|                         | Live<br>stream  | http://xxx.liveplay.myqcloud.com/xxx.m3u8   | Supported     | Supported           |
| HLS(MSUO)               | VOD<br>playback | http://xxx.vod.myqcloud.com/xxx.m3u8        | Supported     | Supported           |
|                         | Live<br>stream  | http://xxx.liveplay.myqcloud.com/xxx.flv    | Supported     | Partially supported |
| ΓLV                     | VOD<br>playback | http://xxx.vod.myqcloud.com/xxx.flv         | Supported     | Partially supported |
| WebRTC                  | Live<br>stream  | webrtc://xxx.liveplay.myqcloud.com/live/xxx | Supported     | Supported           |

### Supported protocols



### Note:

Only H.264 encoding is supported.

The player is compatible with mainstream browsers and can automatically select the optimal playback scheme depending on the browser.

In some browser environments, HLS and FLV video playback depends on Media Source Extensions.

If a browser does not support WebRTC, a WebRTC URL passed in will be converted automatically to better support playback.

## **Supported Features**

| Feature\\Browser                  | Chrome   | Firefox | Edge | QQ<br>Browser | Mac<br>Safari | iOS<br>Safari | WeChat | Android<br>Chrome |
|-----------------------------------|----------|---------|------|---------------|---------------|---------------|--------|-------------------|
| Player dimension configuration    | 1        | 1       | 1    | 1             | 1             | 1             | 1      | 1                 |
| Resuming<br>playback              | 1        | 1       | 1    | ✓             | 1             | 1             | 1      | 1                 |
| Playback speed change             | 1        | 1       | 1    | ✓             | 1             | 1             | 1      | 1                 |
| Preview<br>thumbnails             | 1        | 1       | 1    | ✓             | -             | -             | -      | -                 |
| Changing `fileID`<br>for playback | 1        | 1       | 1    | 1             | 1             | 1             | 1      | 1                 |
| Flipping videos                   | 1        | 1       | 1    | 1             | 1             | 1             | 1      | 1                 |
| Progress bar<br>marking           | <i>√</i> | 1       | 1    | ✓             | 1             | -             | -      | -                 |
| HLS adaptive bitrate              | <b>√</b> | 1       | 1    | ✓             | 1             | 1             | ✓      | 1                 |
| Referer hotlink protection        | <b>√</b> | 1       | 1    | ✓             | 1             | 1             | ✓      | -                 |
| Definition change notifications   | 1        | 1       | 1    | 1             | -             | -             | -      | 1                 |
| Preview                           | 1        | 1       | 1    | 1             | 1             | 1             | 1      | 1                 |
|                                   |          |         |      |               |               |               |        |                   |

S Tencent Cloud

| Playing HLS<br>videos encrypted<br>using standard<br>schemes  | ~        | <i>✓</i> | J  | 1            | 1 | 1 | ✓<br>               | ✓        |
|---|----------|----------|----|--------------|---|---|---------------------|----------|
| Playing HLS<br>videos encrypted<br>using private<br>protocols | 1        | <i>J</i> | 1  | -            | - | - | Android:√<br>iOS: - | 1        |
| Video statistics  | 1        | 1        | 1  | 1            | - | - | -                   | -        |
| Video data<br>monitoring                                      | <i>√</i> | 1        | 1  | <i>✓</i>     | - | - | -                   | -        |
| Custom UI<br>messages   | <i>√</i> | 1        | 1  | <i>✓</i>     | 1 | 1 | ✓                   | <i>s</i> |
| Custom UI   | 1        | 1        | 1  | 1            | 1 | 1 | 1                   | 1        |
| On-screen comments  | <i>√</i> | 1        | 1  | <i>✓</i>     | 1 | 1 | ✓                   | <i>s</i> |
| Watermark   | 1        | 1        | 1  | $\checkmark$ | 1 | 1 | 1                   | 1        |
| Ghost watermark   | 1        | 1        | 1  | 1            | 1 | 1 | 1                   | 1        |
| Playlist  | 1        | 1        | 1  | 1            | 1 | 1 | 1                   | 1        |
| Frame sync<br>under poor<br>network<br>conditions             | 1        | 1        | s. | 1            | 1 | 1 | 1                   | ✓        |

### Note:

Only H.264 encoding is supported.

Chrome and Firefox for Windows and macOS are supported.

Chrome, Firefox, Edge, and QQ Browser need to load hls.js to play HLS.

The Referer hotlink protection feature is based on the Referer field of HTTP request headers. Some HTTP requests initiated by Android browsers do not carry the Referer field.

The player is compatible with mainstream browsers and can automatically select the optimal playback scheme depending on the browser used. For example, for modern browsers such as Chrome, the player uses the HTML5 technology for playback, and for mobile browsers, it uses the HTML5 technology or the browser's built-in capabilities.



## Preparations

From version 5.0.0, the TCPlayer SDK for Web (TCPlayer) requires access to a License authorization for use. If you don't need the new premium functions, you can apply for a basic License to **try TCPlayer for free;** if you want to use the newly added premium functions, you need to purchase a premium License. The detailed information is as follows:

| TCPlayer<br>feature  | Feature Scope   | Required License                      | Pricing                             | Authorization Unit   |
|----------------------|---|---------------------------------------|-------------------------------------|--|
| Basic<br>Functions   | Includes all features<br>provided in versions<br>prior to 5.0.0, see<br>Product Features for<br>details | Web Player Basic<br>Version License   | <b>0 CNY</b><br>Free<br>Application | An exact domain (1<br>License can<br>authorize up to 10<br>exact domains)  |
| Premium<br>Functions | Basic Version<br>Features, VR<br>Playback, Security<br>Check  | Web Player Premium<br>Version License | 399<br>CNY/month<br>Buy Now         | Wildcard Domain (1<br>License can<br>authorize up to 1<br>wildcard domain) |

### Note:

1. Web Player Basic Version License can be applied for free, with a default validity of 1 year; if the remaining validity period is less than 30 days, it can be renewed for free.

2. To facilitate local development, the player won't authenticate localhost or 127.0.0.1; hence, these types of local service domain names need not be applied for when requesting a License.

## Integration Guide

By following these steps, you can add a video player to your website.

### Step 1. Import files into the page

The Player SDK supports two integration methods: CDN and NPM:

### 1. Integration through CDN

Create a new index.html file in your local project and import the player style and script files into the HTML page.

It is recommended to deploy resources on your own when using the Player SDK, click Download player resources.

Deploy the unzipped folder without altering its directory structure to prevent cross-referencing issues between resources.

If the deployment address is aaa.xxx.ccc , import the player style and script files at the appropriate places. When



deploying on your own, you need to manually reference the dependency files under the libs folder of the resource package, otherwise, the Tencent Cloud CDN files will be requested by default.



```
<link href="aaa.xxx.ccc/tcplayer.min.css" rel="stylesheet"/>
<!--To play HLS format videos in modern browsers like Chrome and Firefox through H
<script src="aaa.xxx.ccc/libs/hls.min.x.xx.m.js"></script>
<!--Player script file-->
<script src="aaa.xxx.ccc/tcplayer.vx.x.x.min.js"></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script
```

### 2. Integration through npm



First, install the tcplayer npm package:



npm install tcplayer.js

Import the SDK and style files:



```
import TCPlayer from 'tcplayer.js';
import 'tcplayer.js/dist/tcplayer.min.css';
```

### Step 2. Place the player container

Add the player container to the location on the page where the player is to be displayed. For example, add the following code in index.html (the container ID and dimensions can be custom defined).





<video id="player-container-id" width="414" height="270" preload="auto" playsinline </video>

### Note:

The player container must be a <video> tag.

In the example, player-container-id is the ID of the player container, which you can set yourself.

We recommend you set the size of the player container zone through CSS, which is more flexible than the attribute and can achieve effects such as fit to full screen and container adaption.

The preload attribute in the example specifies whether to load the video after the page is loaded. It is usually set to auto for faster video playback. Other options are: meta (to only load metadata after the page loads) and none (to not load the video after the page loads). Videos will not automatically load on mobile devices due to system restrictions.

The attributes playsinline and webkit-playsinline are used to achieve inline playback in standard mobile browsers without hijacking video playback. This is just an example, please use as needed. Setting the x5-playsinline attribute in the TBS kernel will utilize the X5 UI player.

### Step 3. Initialize the player

After page initialization, you can play video resources. The player supports both video on demand (VOD) and live streaming playback scenarios as follows:

VOD playback: The player can play Tencent Video on Demand media resources through FileID. For the specific VOD process, please refer to the Using the Player for Playback document.

Live playback: The player can pull a live audio/video stream for playback by passing in a URL. For information on generating a Tencent Cloud Streaming Services URL, see Splicing Live Streaming URLs.

URL playback (VOD and live)

File ID playback (VOD)

After page initialization, call the method in the player instance to pass in the URL to the method.



```
// `player-container-id` is the player container ID, which must be the same as in H
var player = TCPlayer('player-container-id', {
    sources: [{
        src: 'Please replace with your playback URL',
        }],
        licenseUrl: 'Please replace with your licenseUrl', // License URL, see the prep
        language: 'Please replace with your Setting language', // Setting language en |
    });
// player.src(url); // URL playback address
```

### ठ Tencent Cloud

In the initialization code on the index.html page, add the following initialization script. Pass in the video ID and appID obtained during the preparation phase from the fileID (in Media Management) and (peek in Account Information > Basic Information).



var player = TCPlayer('player-container-id', { // player-container-id is the player fileID: 'Please enter your fileID', // Enter the fileID of the video to be play appID: 'Please enter your appID', // Enter the appID of your VOD account // Enter the player Signature psign, for information on the Signature and how t psign: 'Please enter your player Signature psign', licenseUrl: 'Please enter your licenseUrl', // Refer to the preparation section language: 'Please replace with your Setting language', // Setting language en | });

### **Please Note:**

Not all videos can be played successfully in a browser. We recommend you use Tencent Cloud's services to transcode a video before playing it.

### Step 4. Implement more features

You can utilize the server-side capabilities of Video on Demand (VOD) for advanced features, such as automatic switching of adaptive streams, video thumbnail previews, and adding video marker information. These features are detailed in Play back a long video, which you can refer to for implementation.

Additionally, the player offers more features. For a list of features and instructions on how to use them, please see the Feature Demonstration page.

# **TCPlayer Resolution Configuration Guide**

Last updated : 2024-05-13 17:49:25

During playback, the video resolution can be switched automatically or manually to accommodate different sizes of playback devices and network conditions, thus enhancing viewing experience. This document will explain several scenarios.

## Live Streaming

Live streaming videos are played in the form of URLs. When initializing the player, you can specify the URL to be played through the sources field. Alternatively, after initializing the player, you can play by calling the src method on the player instance.

### 1. Adaptive Bitrate (ABR)

ABR URLs can seamlessly transition during switches without causing interruptions or jumps, ensuring a smooth transition in both visual and auditory experiences. This technology is also relatively simple to use; it passes the playback URL to the player, which will automatically parse the sub-streams and render the resolution switching component on the control bar.

### Example 1: Playing HLS ABR URLs

During the player's initialization, when receiving an ABR URL, the player will automatically generate a resolution switching component, so it can switch automatically based on network conditions.





```
const player = TCPlayer('player-container-id', { // player-container-id is the play
sources: [{
    src: 'https://hls-abr-url', // hls ABR URL
  }],
});
```

### Note:

Parsing the substreams of an HLS ABR requires the dependency on the MSE API of the playback environment. In browsers not supporting MSE (e.g., Safari on iOS), the browser internally handles this by automatically switching resolutions based on network conditions, but it won't be able to parse multiple resolutions for manual switching.

### Example 2: Playing WebRTC ABR URLs

In the WebRTC ABR scenario, when receiving an URL, the player will automatically decompose the substream URLs based on the ABR template in the URL.



```
const player = TCPlayer('player-container-id',{
  sources: [{
    src: 'webrtc://global-lebtest-play.myqcloud.com/live/lebtest?txSecret=f22a813b2
  }],
  webrtcConfig: {
    // Whether to render multiple resolutions; enabled by default; optional
```



```
enableAbr: true,
// The label name corresponding to the template name; optional
abrLabels: {
    d1080p: 'FHD',
    d540p: 'HD',
    d360p: 'SD',
    auto: 'AUTO',
    },
},
```

The following explanations are provided for the parameters in the WebRTC URL:

1. tabr\_bitrates specifies the ABR template, and the number of templates will determine the number of rendered resolutions. If no separate resolution label is set, the template name (e.g., d1,080p) will be used as the name of the resolution.

2. tabr\_start\_bitrate specifies the initial resolution setting.

3. tabr\_control sets whether automatic resolution switching is enabled. When the automatic switching is enabled, the player will provide an option for automatic resolution adjustment.

### 2. Manually Setting Resolution

If the playback URL is not an ABR URL, you can also manually set the resolution. See the following code:




```
const player = TCPlayer('player-container-id', { // player-container-id is the play
multiResolution:{
    // Configure multiple resolution URLs
    sources:{
        'SD':[{
            src: 'http://video-sd-url',
        }],
        'HD':[{
            src: 'http://video-hd-url',
        }],
        'FHD':[{
```

```
src: 'http://video-fhd-url',
}]
},
// Configure the tag for each resolution
labels:{
    'SD':'Standard Definition', 'HD':'High Definition', 'FHD':'Full High Definition
    },
    // Configure the order of resolutions in the player component
    showOrder:['SD', 'HD', 'FHD'],
    // Configure the default resolution
    defaultRes: 'SD',
},
});
```

## VOD

For VOD playback using a fileID, the player signature specifies the type of file to be played (source, transcoded, or ABR) and the substream resolutions of ABR files. For a complete understanding of the VOD playback process, you can refer to the guide Play back an ABR streaming video.

When calculating the player signature, you can set the display names of substreams of different resolutions through the resolutionNames in the contentInfo field. If you leave it blank or fill in an empty array, the default configuration will be used.



```
resolutionNames: [{
    MinEdgeLength: 240,
    Name: '240P',
}, {
    MinEdgeLength: 480,
    Name: '480P',
}, {
    MinEdgeLength: 720,
    Name: '720P',
}, {
    MinEdgeLength: 1080,
```

```
Name: '1080P',
}, {
MinEdgeLength: 1440,
Name: '2K',
}, {
MinEdgeLength: 2160,
Name: '4K',
}, {
MinEdgeLength: 4320,
Name: '8K',
}]
```

The number of substreams during playback depends on the number of substreams converted according to different ABR templates during transcoding. These substreams will fall within the MinEdgeLength range set by resolutionNames based on short-side length and then be displayed with the corresponding Name as the resolution name.

To get a quick start on generating player signatures, you can use the Tencent Cloud VOD console's Player Signature Generation Tools.

# TCPlayer Swift Live Streaming Downgrade Notice

Last updated : 2024-05-13 17:49:25

## **Downgrade Scenarios**

Live Event Broadcasting (LEB) relies on WebRTC technology, requiring both the operating system and browser to support WebRTC.

Currently, the SDK has been tested on the following operating systems and browsers, with the test results as follows.

| Operating System | OS Version | Browser            | Browser Version | Support for<br>Stream Pull |
|------------------|------------|--------------------|-----------------|----------------------------|
| Windows          | Windows 10 | Chrome             | 86+             | ✓                          |
|                  |            | Firefox            | 88+             | ✓                          |
|                  |            | Microsoft Edge     | 86+             | ✓                          |
| macOS            | 10.5+      | Safari             | 13.1+           | ✓                          |
|                  |            | Chrome             | 86+             | $\checkmark$               |
|                  |            | Firefox            | 88+             | ✓                          |
|                  |            | Microsoft Edge     | 86+             | $\checkmark$               |
| iOS              | 13.1.1+    | Safari             | 13.7+           | ✓                          |
|                  |            | Chrome             | 86+             | ✓                          |
|                  |            | Firefox            | 33+             | ✓                          |
|                  |            | Microsoft Edge     | 89              | $\checkmark$               |
|                  |            | WeChat<br>embedded | -               | 1                          |
| Android          | -          | Chrome             | 86+             | ✓                          |
|                  |            | Firefox            | 88+             | ✓                          |
|                  |            | WeChat<br>embedded | X5 kernel       | 1                          |



| WeChat   | XWeb kernel | $\checkmark$ |
|----------|-------------|--------------|
| embedded |             |              |

Additionally, in some browsers that support WebRTC, there may be decoding failures or server-side issues. In these cases, the player will convert the WebRTC URL to a more compatible HLS URL for playback. This behavior is known as downgrade processing.

To summarize, there are several scenarios that trigger downgrading:

The browser environment does not support WebRTC.

Failed to connect to the server, and the number of retries has exceeded the preset value (internal status code -2004). Failed to decode during playback (internal status code -2005).

Other WebRTC-related errors (internal status code -2001).

## Downgrade Methods

#### 1. Automatic Downgrade

During player initialization, the LEB URL is received through the sources field. In environments requiring downgrade processing, the player automatically converts the LEB URL to the HLS URL. For example, the LEB URL





webrtc://global-lebtest-play.myqcloud.com/live/lebtest?txSecret=f22a813b284137ed10d

will automatically convert to:





https://global-lebtest-play.myqcloud.com/live/lebtest.m3u8?txSecret=f22a813b284137e

#### 2. Specified Downgrade

In ABR playback scenarios, downgrade processing requires manually specifying an HLS URL. Direct format conversion is not supported. This method can also be used for scenarios where users require manual configuration of downgrade URLs, and it is compatible with protocols beyond HLS.





```
var player = TCPlayer('player-container-id',{
    sources: 'webrtc://global-lebtest-play.myqcloud.com/live/lebtest?txSecret=f22a8
    webrtcConfig: {
        fallbackUrl: 'https://global-lebtest-play.myqcloud.com/live/lebtest_HLSABR.m3
    },
});
```

## Downgrade Callback

When a downgrade is triggered, the player will initiate a callback.



```
player.on('webrtcfallback', function(event) {
    console.log(event);
}
```

});

## iOS Integration Guide

Last updated : 2024-04-18 17:06:53

## Overview

The iOS Player is an open-source component that allows you to integrate powerful playback capabilities similar to those in Tencent Video into your project with just a few lines of code changes. In addition to basic features such as landscape/portrait orientation, resolution selection, gestures, and small-window playback, it also supports buffering, software/hardware decoding, and changing playback speed. Compared with built-in players, the RT-Cube Player supports more formats, has better compatibility, and offers more capabilities. It also features instant streaming and low latency, and comes with advanced features such as thumbnail generation.

If the Player component cannot meet your requirements, and you have some knowledge of engineering, you can integrate the Player SDK to customize the UI and playback features.

### Limits

- 1. Activate VOD. If you don't have an account yet, sign up for one first.
- 2. Download and install Xcode from App Store.
- 3. Download and install CocoaPods as instructed at the CocoaPods website.

## This Document Describes

How to integrate the Player component for iOS How to create and use a player

## Prerequisites

#### Step 1. Download the player code package

GitHub page: LiteAVSDK/Player\_iOS You can download a ZIP file of the Player component from the GitHub page or use the Git clone command to download the component. Download the ZIP file Download using Git command



Go to the GitHub page and click **Code > Download ZIP**.



1. First, make sure that your computer has Git installed; if not, you can install it as instructed in Git Installation Tutorial.

2. Run the following command to clone the code of the Player component to your local system.





git clone git@github.com:tencentyun/SuperPlayer\_iOS.git

If you see the following information, the project code has been cloned to your local system successfully.





Cloning to 'SuperPlayer\_iOS'... remote: Enumerating objects: 2637, done. remote: Counting objects: 100% (644/644), done. remote: Compressing objects: 100% (333/333), done. remote: Total 2637 (delta 227), reused 524 (delta 170), pack-reused 1993 Receiving the object: 100% (2637/2637), 571.20 MiB | 3.94 MiB/s, done. Processing delta: 100% (1019/1019), done.

Below is the directory structure of the component's source code after decompression:

Filename

Description

| SDK             | The folder of the Player component's frameworks and static libraries. |
|-----------------|---|
| Demo            | The folder of the Player demo.  |
| Арр             | The entry point UI.   |
| SuperPlayerDemo | The Player demo.  |
| SuperPlayerKit  | The Player component.   |

#### Step 2. Integrate the component

This step describes how to integrate the Player component. We recommend you integrate it through CocoaPods or

manually download the SDK and then import it into your current project.

Integrate via CocoaPods

Manually download the SDK

1. To install the component using CocoaPods, add the code below to Podfile:

(1) Directly integrate SuperPlayer as a Pod:



pod 'SuperPlayer

To use the Player edition, add the following dependency to  $\ensuremath{\left| \ensuremath{\mathsf{podfile}} \ensuremath{\right|}}$  :



pod 'SuperPlayer/Player'

To use the Player Premium edition, add the following dependency to  ${\tt podfile}$  :



pod 'SuperPlayer/Player\_Premium'

To use the All-in-one edition, add the following dependency to  ${\tt podfile}$  :





pod 'SuperPlayer/Professional'

- $2.\,Run$  pod install or pod update .
- 1. Download the SDK and demo at GitHub.
- 2. Import TXLiteAVSDK\_Player\_Premium.framework into your project and select **Do Not Embed**.
- 3. Copy Demo/TXLiteAVDemo/SuperPlayerKit/SuperPlayer to your project directory.
- 4. The third-party libraries the player depends on are AFNetworking , SDWebImage , Masonry , and TXLiteAVSDK\_Player .

To integrate TXLiteAVSDK\_Player manually, you need to add the required system frameworks and libraries:



System frameworks: MetalKit, ReplayKit, SystemConfiguration, CoreTelephony, VideoToolbox, CoreGraphics,

AVFoundation, Accelerate, and MobileCoreServices.

System libraries: libz, libresolv, libiconv, libc++, and libsqlite3.

For detailed directions, see Manually integrate the SDK.

In addition, you need to add TXFFmpeg.xcframework and TXSoundTouch.scframework under the TXLiteAVSDK\_Player file as dynamic libraries.

| ✓ Frameworks, Libraries, and Embedded Content |                                |                |  |  |  |
|---|--------------------------------|----------------|--|--|--|
|   | Name                           | Embed          |  |  |  |
|   | Iibc++.tbd                     |                |  |  |  |
|   | 🖹 libiconv.tbd                 |                |  |  |  |
|   | libPods-SuperPlayerDemo.a      |                |  |  |  |
|   | Ibresolv.tbd                   |                |  |  |  |
|   | 🖹 libsqlite3.tbd               |                |  |  |  |
|   | 🖹 libz.tbd                     |                |  |  |  |
|   | 🖴 MetalKit.framework           | Do Not Embed 🗘 |  |  |  |
|   | 🚘 ReplayKit.framework          | Do Not Embed 💲 |  |  |  |
|   | 🚔 TXFFmpeg.xcframework         | Embed & Sign 💲 |  |  |  |
|   | A TXLiteAVSDK_Player.framework | Do Not Embed 🗘 |  |  |  |
|   | 🚘 TXSoundTouch.xcframework     | Embed & Sign 🗘 |  |  |  |
|   | 🚘 VideoToolbox.framework       | Do Not Embed 🗘 |  |  |  |
|   | + -                            |                |  |  |  |

5. If you integrate TXLiteAVSDK\_Player as a pod, no libraries need to be added.

#### Step 3. Use the player features

This step describes how to create a player and use it for video playback.

#### 1. Create a player

Create a SuperPlayerView object to play videos ( SuperPlayerView is the main class of the player).





```
// Import the header file
#import <SuperPlayer/SuperPlayer.h>
```

```
// Create a player
_playerView = [[SuperPlayerView alloc] init];
// Set a delegate for events
_playerView.delegate = self;
// Set the parent view. _playerView will be automatically added under holderView.
_playerView.fatherView = self.holderView;
```

#### 2. License configuration

If you have obtained a license, you can view the license URL and key in the VOD console.

If you don't have the required license yet, you can get it as instructed in Video Playback License.

After obtaining the License information, you need to initialize and configure the License before calling the relevant interfaces of the SDK. For detailed tutorials, please see Configuring View License.

#### 3. Video playback:

This step describes how to play back a video. The Player for iOS supports playback through FileId in VOD or

URL. We recommend you **integrate the FileId** because it allows you to use more VOD capabilities.

Play by VOD file ID

Play by URL

Play Local video

A video file ID is returned by the server after the video is uploaded.

1. After a video is published from a client, the server will return a file ID to the client.

2. After a video is uploaded to the server, the notification for successful upload will contain a file ID for the video.

If the video you want to play is already saved with VOD, you can go to Media Assets to view its file ID.

| Video Name/ID                               | Video Status | Video Cate <b>T</b> | Uploading 🗘            | Expiration Time | Storage             |
|---|--------------|---------------------|------------------------|-----------------|---------------------|
| test_2022-04-15-17<br>ID:387702299327461625 | 🔗 Normal     | Other               | 2022-04-15<br>17:47:24 | Permanent       | Outside<br>mainlanc |
| test_2022-04-07-16<br>ID:387702298823695918 | 🔗 Normal     | Other               | 2022-04-07<br>16:14:00 | Permanent       | Outside<br>mainlanc |
| test_2022-04-07-16<br>ID:387702298823695507 | 🕑 Normal     | Other               | 2022-04-07<br>16:12:07 | Permanent       | Outside<br>mainlanc |

#### Note:

To play by VOD file ID, you need to use the Adaptive-HLS template (ID: 10) to transcode the video or use the player signature psign to specify the video to play; otherwise, the playback may fail. For more information on how to transcode a video and generate psign, see Play back a video with the Player component and Player Signature.
 If a "no v4 play info" error occurs, it indicates that you haven't transcoded the video or used the player signature correctly. Troubleshoot the issue according to the above documents or get the playback URL of the video and play it by URL.

3. We recommend you transcode videos for playback because untranscoded videos may experience compatibility issues during playback.





```
// If you haven't enabled hotlink protection and a "no v4 play info" error occurs,
SuperPlayerModel *model = [[SuperPlayerModel alloc] init];
model.appId = 1400329071;// Configure AppId
model.videoId = [[SuperPlayerVideoId alloc] init];
model.videoId.fileId = @"5285890799710173650"; // Configure `FileId`
// `psign` is a player signature. For more information on the signature and how to
model.videoId.pSign = @"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTQwMDMyOT
[_playerView playWithModelNeedLicence:model];
```





```
SuperPlayerModel *model = [[SuperPlayerModel alloc] init];
model.videoURL = @"http://your_video_url.mp4"; // Enter the URL of the video to p
[_playerView playWithModelNeedLicence:model];
```





SuperPlayerModel \*model = [[SuperPlayerModel alloc]init];
//Add your video file to the project, and then get the file path of the video throu
NSString \*filePath = [[NSBundle mainBundle] pathForResource:@"your\_video\_name" ofTy
model.videoURL = [filePath stringByReplacingOccurrencesOfString:@"file://" withStri
[\_playerView playWithModelNeedLicence:model];

#### 4.Stop playback

If the player is no longer needed, call resetPlayer to reset the player and free up memory.



[\_playerView resetPlayer];

At this point, you have learned how to create a player, use it to play videos, and stop playback.

## More Features

#### 1. Full screen playback

The Player component supports full screen playback, where it allows setting screen lock, volume and brightness control through gestures, on-screen commenting, screencapturing, and definition selection. This feature can be tried out in **TCToolkit App** > **Player** > **Player Component**, and you can enter the full screen playback mode by clicking the full screen icon.

You can call the API below to enter full screen from the windowed playback mode:



```
- (void)superPlayerFullScreenChanged:(SuperPlayerView *)player {
    // You can customize the logic after switching to the full screen mode here
}
```



#### Features of full screen playback mode

Back to windowed mode

Enable screen locking

On-screen comments

Screenshot

Change resolution

Tap the back button to return to the windowed mode. The delegate method that will be triggered after the SDK implements the logic for exiting full screen is as follows:



```
// The back button tapping event
- (void)superPlayerBackAction:(SuperPlayerView *)player;
Triggered by tapping of the back button at the top left
// The exit full screen notification
- (void)superPlayerFullScreenChanged:(SuperPlayerView *)player;
```

Screen locking disables touch screen and allows users to enter an immersive playback mode. The SDK will handle the tapping event and no callbacks will be sent.



// Use the API below to enable/disable screen locking
@property(nonatomic, assign) BOOL isLockScreen;



After the on-screen commenting feature is enabled, text comments sent by users will be displayed on the screen. Get the SPDefaultControlView object and, during initialization of the player view, set an event for the onscreen comment button of SPDefaultControlView . The on-screen comment content and view are customized by yourself. For details, see CFDanmakuView , CFDanmakuInfo , and CFDanmaku in SuperPlayerDemo .



SPDefaultControlView \*dv = (SPDefaultControlView \*)\*\*self\*\*.playerView.controlView; [dv.danmakuBtn addTarget:\*\*self\*\* action:\*\*@selector\*\*(danmakuShow:) forControlEven



CFDanmakuView: Configure the attributes of on-screen commenting during initialization.



// The following attributes are required-----// On-screen time
@property(nonatomic, assign) CGFloat duration;
// On-screen time in the center, at top, and at bottom
@property(nonatomic, assign) CGFloat centerDuration;
// On-screen comment line height
@property(nonatomic, assign) CGFloat lineHeight;
// Spacing between on-screen comment lines
@property(nonatomic, assign) CGFloat lineMargin;

```
// Maximum number of on-screen comment lines
@property(nonatomic, assign) NSInteger maxShowLineCount;
// Maximum number of on-screen comment lines in the center, at top, and at bottom
@property(nonatomic, assign) NSInteger maxCenterLineCount;
```

The Player component allows users to take and save a screenshot of a video during playback. The SDK will handle the screenshot button tapping event and no callbacks will be sent for successful or failed screenshots. Screenshots are saved to the phone album.

Users can change the video definition (such as SD, HD, and FHD) during playback. After the definition selection button is tapped, the SDK will implement the logic for displaying the definition selection view and handle the selection event. No callbacks will be sent.

#### 2. Floating window playback

The Player component supports playback in a small floating window, which allows users to switch to another page of the application without interrupting the video playback. You can try out this feature in TCToolkit App > **Player** > **Player Component** by tapping **Back** in the top-left corner.

 $\leftarrow$ 

## Display over other apps

### Allow display over other apps



Allow this app to display on top of other apps you're using. This app will be able to see where you tap or change what's displayed on the screen.





// Tapping the back button during playback in portrait mode will trigger the API
[SuperPlayerWindowShared setSuperPlayer:self.playerView];
[SuperPlayerWindowShared show];
// The API triggered by tapping the floating window to return to the main window
SuperPlayerWindowShared.backController = self;

#### 3. Thumbnail

The Player component supports customizing a video thumbnail, which is displayed before the callback is received for playing back the first video frame. This feature can be tried out in **TCToolkit App** > **Player** > **Player Component** >



#### Thumbnail Customization Demo.

When the Player component is set to the automatic playback mode PLAY\_ACTION\_AUTO\_PLAY, the thumbnail will be displayed before the first video frame is loaded.

When the Player component is set to the manual playback mode <u>PLAY\_ACTION\_MANUAL\_PLAY</u>, videos are played only after users tap the play button, and the thumbnail will be displayed until the first video frame is loaded. You can set the thumbnail by specifying the URL of a local or online file. For detailed directions, see the code below. If you play by VOD file ID, you can also set the thumbnail in the VOD console.



```
SuperPlayerModel *model = [[SuperPlayerModel alloc] init];
SuperPlayerVideoId *videoId = [SuperPlayerVideoId new];
```



```
videoId.fileId = @"8602268011437356984";
model.appId = 1400329071;
model.videoId = videoId;
// Playback mode, which can be set to automatic (`PLAY_ACTION_AUTO_PLAY`) or manual
model.action = PLAY_ACTION_MANUAL_PLAY;
// Specify the URL of an online file to use as the thumbnail. If `coverPictureUrl`
model.customCoverImageUrl = @"http://1500005830.vod2.myqcloud.com/6c9a5118vodcq1500
[self.playerView playWithModelNeedLicence:model];
```

#### 4. Video playlist loop

The Player component supports looping video playlists.

After a video ends, the next video in the list can be played automatically or users can manually start the next video. After the last video in the list ends, the first video in the list will start automatically.

You can try out this feature in **TCToolkit App** > **Player** > **Player Component** > **Video List Loop Demo**.


```
// Step 1. Create a `NSMutableArray` for the loop data
NSMutableArray *modelArray = [NSMutableArray array];
SuperPlayerModel *model = [SuperPlayerModel new];
SuperPlayerVideoId *videoId = [SuperPlayerVideoId new];
videoId.fileId = @"8602268011437356984";
model.appId = 1252463788;
model.videoId = videoId;
[modelArray addObject:model];
model = [SuperPlayerModel new];
```

```
videoId = [SuperPlayerVideoId new];
```



```
videoId.fileId = @"4564972819219071679";
model.appId = 1252463788;
model.videoId = videoId;
[modelArray addObject:model];
// Step 2. Call the loop API of `SuperPlayerView`
```

[self.playerView playWithModelListNeedLicence:modelArray isLoopPlayList:YES startIn



(void)playWithModelListNeedLicence:(NSArray \*)playModelList isLoopPlayList:(BOOL)is

API parameters:

| Parameter     | Туре      | Description   |  |
|---------------|-----------|---|--|
| playModelList | NSArray * | Loop data list                                      |  |
| isLoop        | Boolean   | Whether to loop the playlist                        |  |
| index         | NSInteger | Index of the video from which to start the playback |  |

#### 5. Picture-in-Picture (PiP) feature

The Picture-in-Picture (PiP) feature has been launched on iOS 9 but can currently be used only on iPads. To use PiP on an iPhone, you need to update the iOS version to iOS 14.

The Player component supports both in-app PiP and system-wide PiP. To use the feature, you need to enable background modes: In Xcode, choose your target, click **Signing & Capabilities** > **+Capability** > **Background Modes**, and select **Audio**, **AirPlay**, **and Picture in Picture**.

| Background Modes     Background Modes     Section     Sect |   |
|--|---|
|  | Modes 🗹 Audio, AirPlay, and Picture in Picture<br>Location updates<br>Voice over IP |

Code sample for using PiP capabilities:



```
// Enter the PiP mode
if (![TXVodPlayer isSupportPictureInPicture]) {
    return;
}
[_vodPlayer enterPictureInPicture];
// Exit the PiP mode
[_vodPlayer exitPictureInPicture];
```

#### 6. Video preview

The Player component supports video preview, which is useful if you want to allow non-subscribers to watch the beginning of a video. We offer parameters for you to set the video preview duration, pop-up message, and preview end screen. You can find a demo for this feature in the TCToolkit app: **Player Component > Preview Feature Demo**.



```
// Step 1. Create a preview model
TXVipWatchModel *model = [[TXVipWatchModel alloc] init];
model.tipTtitle = @"You can preview 15 seconds of the video. Become a subscriber t
model.canWatchTime = 15;
// Step 2. Set the preview model
self.playerView.vipWatchModel = model;
```

// Step 3. Call the method below to display the preview
[self.playerView showVipTipView];

TXVipWatchModel class parameter description:

| Parameter    | Туре     | Description                 |
|--------------|----------|-----------------------------|
| tipTtitle    | NSString | Pop-up message              |
| canWatchTime | float    | Preview duration in seconds |

#### 7. Dynamic watermark

The Player component allows you to add a randomly moving text watermark to protect your content against piracy. Watermarks are visible in both the full screen mode and windowed mode. The text, font size, and color of a watermark are customizable. You can find a demo for this feature in the TCToolkit app: **Player > Player Component > Dynamic Watermark Demo**.



```
// Step 1. Create a video source information model
SuperPlayerModel * playermodel = [SuperPlayerModel new];
// Add other information of the video source
// Step 2. Create a dynamic watermark model
DynamicWaterModel *model = [[DynamicWaterModel alloc] init];
// Step 3. Set the data of the dynamic watermark
model.dynamicWatermarkTip = @"shipinyun";
model.textFont = 30;
model.textColor = [UIColor colorWithRed:255.0/255.0 green:255.0/255.0 blue:255.0/255
playermodel.dynamicWaterModel = model;
// Step 4. Call the method below to display the dynamic watermark
```

[self.playerView playWithModelNeedLicence:playermodel];

Parameters for DynamicWaterModel :

| Parameter           | Туре     | Description    |
|---------------------|----------|----------------|
| dynamicWatermarkTip | NSString | Watermark text |
| textFont            | CGFloat  | Font size      |
| textColor           | UIColor  | Text color     |

#### 8. Video download

Video download allows users to cache online videos and watch them offline. The cached video can be played back only in the client but cannot be actually downloaded to the device. This feature can effectively prevent downloaded videos from being distributed without authorization and protect the video security.

You can try out this feature in full screen mode in TCToolkit App > Player > Player Components > Offline Cache.



VideoCacheView (cache selection list view) is used to select and download videos at different definitions. After selecting the definition in the top-left corner, click the option of the video to be downloaded. When a check mark appears, the download has started. After clicking the **video download list** button below, you will be redirected to the Activity where VideoDownloadListView is located.





// Step 1. Initialize the cache selection list view
//@property (nonatomic, strong) VideoCacheView \*cacheView;
\_cacheView = [[VideoCacheView alloc] initWithFrame:CGRectZero];
\_cacheView.hidden = YES;
[self.playerView addSubview:\_cacheView];

// Step 2. Set the options of the video being played back
[\_cacheView setVideoModels:\_currentPlayVideoArray currentPlayingModel:player.player

// Click event of the \*\*video download list\*\* button
- (UIButton \*)viewCacheListBtn;



#### - (void)setVideoModels:(NSArray \*)models currentPlayingModel:(SuperPlayerModel \*)cu

#### API parameters:

| Parameter        | Туре         | Description   |
|------------------|--------------|---|
| models           | NSArray      | The video data model of the download list           |
| SuperPlayerModel | currentModel | The video data model of the video being played back |

VideoCacheListView (video download list)

displays the list of views of all the videos that are being downloaded and have been downloaded.

When this button is clicked :

if the download is in progress, it will be paused.

if it is paused, it will be resumed.

if it has completed, the video will be played back.







```
// Add data, which is obtained from the `TXVodDownloadManager#getDownloadMediaInfoL
NSArray<TXVodDownloadMediaInfo *> *array = [[[TXVodDownloadManager shareInstance] g
for (TXVodDownloadMediaInfo *info in array) {
    VideoCacheListModel *model = [[VideoCacheListModel alloc] init];
    model.mediaInfo = info;
    [self.videoCacheArray addObject:model];
}
// List items support operations such as click to play and hold and press to delete
```

- (void)longPress:(UILongPressGestureRecognizer \*)longPress; // Hold and press



The downloaded video supports playing without network connection, please refer to the following code when playing:



```
NSArray<TXVodDownloadMediaInfo *> *mediaInfoList = [[TXVodDownloadManager shareInst
TXVodDownloadMediaInfo *mediaInfo = [mediaInfoList firstObject];
SuperPlayerUrl *superPlayerUrl = [[SuperPlayerUrl alloc] init];
superPlayerUrl.title = @"*******";
superPlayerUrl.url = mediaInfo.playpath;
NSArray<SuperPlayerUrl *> *multiVideoURLs = @[superPlayerUrl];
SuperPlayerModel *playerModel = [[SuperPlayerModel alloc] init];
playerModel.multiVideoURLs = multiVideoURLs;
[self.playerView playWithModelNeedLicence:playerModel];
```



#### Note:

When video files are downloaded without network playback, be sure to obtain the download list and play through the PlayPath of the video object TXVodDownloadMediaInfo in the download list, and do not directly save the PlayPath object.

#### 9. Image sprite and timestamp information

#### **Timestamp information**

You can add text descriptions at key positions on the progress bar, which the user can click to view and quickly understand the video information at the current position. After clicking the video information, the user can seek to the desired position.

You can try out this feature in full screen mode in TCToolkit App > Player > Player Components > Tencent Cloud Video.



#### Image sprite

Users can view video thumbnails when dragging or seeking on the progress bar so as to quickly understand the video content at the specified position. The thumbnail preview is implemented based on the video's image sprite. You can generate the image sprite of a video file in the VOD console or directly generate an image sprite file. You can try out this feature in full screen mode in TCToolkit App > Player > Player Components > Tencent Cloud Video.









// Step 1. Get the image sprite and timestamp information in the `onPlayEvent` call
[self.playerView playWithModelNeedLicence:playerModel];

// Step 2. Get keyframes and image sprite information in the `VOD\_PLAY\_EVT\_GET\_PLAY
NSString \*imageSpriteVtt = [param objectForKey:VOD\_PLAY\_EVENT\_IMAGESPRIT\_WEBVTTURL]
NSArray<NSString \*> \*imageSpriteList = [param objectForKey:VOD\_PLAY\_EVENT\_IMAGESPRI
NSArray<NSURL \*> \*imageURLs = [self convertImageSpriteList:imageSpriteList];
[self.imageSprite setVTTUrl:[NSURL URLWithString:imageSpriteVtt] imageUrls:imageURLs

// Step 3. Display the obtained timestamp information and image sprite on the UI
if (self.isFullScreen) {

```
thumbnail = [self.imageSprite getThumbnail:draggedTime];
}
if (thumbnail) {
   [self.fastView showThumbnail:thumbnail withText:timeStr];
}
```

#### 10. External subtitles

#### Note:

External subtitles depend on the premium version SDK of the media player, and the SDK needs to be version 11.3 or above to support it.



Currently, only SRT and VTT subtitle formats are supported. The usage is as follows:

#### Step 1: Add external subtitles.

Pass the external subtitle category field to SuperPlayerModel#subtitlesArray .





```
// // Pass in subtitle url, subtitle name, and subtitle type
SuperPlayerSubtitles *subtitleModel = [[SuperPlayerSubtitles alloc] init];
subtitleModel.subtitlesUrl = @"https://mediacloud-76607.gzc.vod.tencent-cloud.com/D
subtitleModel.subtitlesName = @"ex-cn-srt";
subtitleModel.subtitlesType = 0;
[subtitlesArray addObject:subtitleModel];
```

# // Play [self.playerView playWithModelNeedLicence:model];

#### Step 2: Switch subtitles after playback.





```
// After the video starts playing, select the added external subtitles.
- (void)controlViewSwitch:(UIView *)controlView withSubtitlesInfo:(TXTrackInfo *)in
if (info.trackIndex == -1) {
    [self.vodPlayer deselectTrack:preInfo.trackIndex];
    self->_lastSubtitleIndex = -1;
} else {
    if (preInfo.trackIndex != -1) {
        // Deselect other subtitles if they are not needed
        [self.vodPlayer deselectTrack:preInfo.trackIndex];
    }
    // Select subtitles. [self.vodPlayer
```

### STencent Cloud

```
[self.vodPlayer selectTrack:info.trackIndex];
self->_lastSubtitleIndex = info.trackIndex;
}
}
```

#### Step 3: Configure subtitle styles.

Subtitle styles can be configured before or during playback.



TXPlayerSubtitleRenderModel \*model = [[TXPlayerSubtitleRenderModel alloc] init]; model.canvasWidth = 1920; // Subtitle render canvas width model.canvasHeight = 1080; // Subtitle render canvas height

### S Tencent Cloud

model.isBondFontStyle = NO; // Set whether the subtitle font is bold model.fontColor = 0xFF000000; // Set the subtitle font color, default white and opa [\_txVodPlayer setSubtitleStyle:model];

#### 11. Ghost watermark

The content of the ghost watermark is filled in the player signature and is ultimately displayed on the playback end through collaboration between the cloud and the player, ensuring the security of the watermark throughout the transmission process. Follow the tutorial to configure the ghost watermark in the player signature. The ghost watermark only appears on the video for a very short time, and this flashing has a minimal impact on viewing the video. The position of each watermark appearance is not fixed, which eliminates attempts by others to cover up the watermark. The effect is shown in the figure below. A watermark appears once when the video starts playing, and then disappears. The content of the ghost watermark can be obtained through [param

objectForKey:@"EVT\_KEY\_WATER\_MARK\_TEXT"] after receiving the VOD\_PLAY\_EVT\_GET\_PLAYINFO\_SUCC event from the player.

Note: Supported from player version 11.6.







```
// Step 1: Configure the FileId that supports ghost watermark to play the video.
SuperPlayerModel *model = [[SuperPlayerModel alloc] init];
model.appId = 1500006438;
model.videoId = [[SuperPlayerVideoId alloc] init];
model.videoId.fileId = @"387702307847129127";
model.videoId.pSign =
@"eyJhbGciOiJIUzI1NiISInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDAwNjQzOCwiZmlsZUlkIjoiMzg3
[_playerView playWithModelNeedLicence:model];
// Step 2: After SuperPlayerView receives the PLAY_EVT_GET_PLAYINFO_SUCC event, obt
```

- (void)onPlayEvent:(TXVodPlayer \*)player event:(int)EvtID withParam:(NSDictionary



```
dispatch_async(dispatch_get_main_queue(), ^{
        if (EvtID == PLAY_EVT_PLAY_EVT_GET_PLAYINFO_SUCCPLAY_PROGRESS) {
            NSString *ghostWaterText = [param objectForKey:@"EVT_KEY_WATER_MARK_TEX
            if (ghostWaterText && ghostWaterText.length > 0) {
                DynamicWaterModel *model = [[DynamicWaterModel alloc] init];
                model.showType = ghost;
                model.duration = self.playerModel.duration;
                model.dynamicWatermarkTip = ghostWaterText;
                model.textFont = 30;
                model.textColor = [UIColor redColor];
                if (![self.subviews containsObject:self.watermarkView]) {
                    [self addSubview:self.watermarkView];
                    [self.watermarkView mas_makeConstraints: ^ (MASConstraintMaker *m
                        make.edges.equalTo(self);
                    }];
                }
                [self.watermarkView setDynamicWaterModel:model];
            }
        }
    };
}
```

# Demo

To try out more features, you can directly run the demo project or scan the QR code to download the TCToolkit App demo.

#### Running a demo project

1. In the Demo directory, run the pod update command to generate the TXLiteAVDemo.xcworkspace file again.

2. Double-click the file to open it, modify the certificate, and run the project on a real device.

3. After the demo is run successfully, go to Player > Player Component to try out the player features.

#### TCToolkit app

You can try out more features of the Player component in TCToolkit App > Player.

During the application upgrade and maintenance, the demo source code can still be used normally.

# Android Integration Guide

Last updated : 2024-04-18 17:06:54

# Overview

The Tencent Cloud RT-Cube Player for Android is an open-source player component of Tencent Cloud. It integrates quality monitoring, video encryption, Top Speed Codec, definition selection, and small window playback and is suitable for all VOD and live playback scenarios. It encapsulates complete features and provides upper-layer UIs to help you quickly create a playback program comparable to mainstream video applications. If the Player component cannot meet your requirements, and you have some knowledge of engineering, you can integrate the Player SDK to customize the UI and playback features.

### Limits

1. To try out all features of the player, we recommend you activate VOD. If you don't have an account yet, sign up for one first. If you don't use the VOD service, you can skip this step; however, you will only be able to use basic player features after integration.

2. Download and install Android Studio. If you have already done so, skip this step.

# This Document Describes

- 1. How to integrate the Player component for Android
- 2. How to create and use the player

# Prerequisites

#### Step 1. Download the player code package

GitHub page: LiteAVSDK/Player\_Android You can download the Player for Android by downloading the Player component ZIP package or running the Git clone command. Download the ZIP file Download using Git command Go to the Player GitHub page and click Code > Download ZIP.

| ᢪ master ▾ 같 ♡ | Go to file                     |
|----------------|--------------------------------|
| Demo           | Clone                          |
| SDK            | Use Git or checkout with SVN u |
| README.md      | 단 Open with GitHub Desk        |
| E README.md    | Download ZIP                   |

1. First, make sure that your computer has Git installed; if not, you can install it as instructed in Git Installation Tutorial.

2. Run the following command to clone the code of the Player component to your local system.





git clone git@github.com:tencentyun/SuperPlayer\_Android.git

If you see the following information, the project code has been cloned to your local system successfully.





Cloning to 'SuperPlayer\_Android'... remote: Enumerating objects: 2637, done. remote: Counting objects: 100% (644/644), done. remote: Compressing objects: 100% (333/333), done. remote: Total 2637 (delta 227), reused 524 (delta 170), pack-reused 1993 Receiving the object: 100% (2637/2637), 571.20 MiB | 3.94 MiB/s, done. Processing delta: 100% (1019/1019), done.

After the project is downloaded, the directory generated after decompression of the source code is as follows:

Filename

Description

| LiteAVDemo(Player)                    | The Player demo project, which can be run directly after being imported into Android Studio.   |  |  |
|---------------------------------------|--|--|--|
| арр                                   | Homepage entry   |  |  |
| superplayerkit                        | The Player component (SuperPlayerView), which provides common features such as playback, pause, and gesture control.   |  |  |
| superplayerdemo                       | The Player component demo code   |  |  |
| common                                | Tool module  |  |  |
| SDK                                   | Player SDK, including LiteAVSDK_Player_x.x.aar (SDK         K       provided in AAR format) and LiteAVSDK_Player_x.x.zip (SDK         provided in lib and JAR formats) |  |  |
| Player Documentation<br>(Android).pdf | The Player component user guide  |  |  |

#### Step 2. Integrate the component

This step describes how to integrate the player. You can integrate the project by using Gradle for automatic loading, manually downloading the AAR and importing it into your current project, or importing the JAR and SO libraries.

Automatic loading in Gradle (AAR)

Manual download in Gradle (AAR)

SDK integration (jar + so)

1. Download the SDK + demo package for Android here.

2. Copy the Demo/superplayerkit module to your project and then configure as follows:

Import superplayerkit into setting.gradle in your project directory.





include ':superplayerkit'

Open the build.gradle file of the superplayerkit project and modify the constant values of compileSdkVersion , buildToolsVersion , minSdkVersion , targetSdkVersion , and rootProject.ext.liteavSdk .





```
compileSdkVersion 26
buildToolsVersion "26.0.2"
defaultConfig {
  targetSdkVersion 23
  minSdkVersion 19
}
dependencies {
  // To integrate an older version, change `latest.release` to the corresponding ve
  implementation 'com.tencent.liteav:LiteAVSDK_Player_Premium:latest.release'
```



```
// If you want to integrate the basic version of the player
// implementation 'com.tencent.liteav:LiteAVSDK_Player:latest.release'
}
```

Import the common module into your project as instructed above and configure it.

3. Configure the mavenCentral library in Gradle, and LiteAVSDK will be automatically downloaded and updated. Open app/build.gradle and configure as follows:

| ect              | 🔲 Project 👻 😨 🔁 🗢 —   | 🗬 build.gradle (:app) $	imes$ | $ ot\!$             |
|------------------|---|-------------------------------|---|
| Proj             | LiteAVDemo(Player) ~/Downloads/LiteA                        | Gradle files have chang       | ed since last project sync. A project sync may be necessary for the IDE to work properly. |
|                  | > idea  |                               | versionCode 24  |
| ger              | <pre>/ .idea / .idea / .idea</pre>                          |                               | versionName "10.8.0.3039"   |
| lana             | > src   |                               |   |
| ce N             | 🛃 .gitignore  |                               |   |
| Inose            | wild.gradle   |                               | multiDexEnabled true  |
| °<br>₽           | proguard-rules.pro  |                               | ndk { NdkOptions it ->  |
|                  | > gradle  |                               | abiFilters "armeabi-v7a", "arm64-v8a" 3   |
|                  | > 🖬 superplayerdemo   |                               | ۰<br>۲  |
|                  | > 🖬 superplayerkit  |                               | 1   |
|                  | <ul> <li>build.gradle</li> <li>gradle properties</li> </ul> | 18 - }                        |   |
|                  | and gradlew   |                               |   |
|                  | 🛔 gradlew.bat   | 20 🗄 sig                      | ningConfigs {}  |
|                  | local.properties  | 24 🗄 bui                      | ldTypes {}  |
|                  | W settings.gradie   | 31 m nac                      | kagingOntions {   |
| Í                | Scratches and Consoles                                      |                               |   |
|                  |   | 38 🖃 dex                      | Uptions {}  |
| ture             |   | 42 ± com                      | pileOptions {}  |
| itruc            |   |                               |   |
|                  |   | 47                            |   |
| ss               |   |                               | noion (   |
| vorite           |   | 48 – depende                  |   |
| Fa               |   |                               |   |
| Build Variants 🗙 |   | 50 imp                        | <pre>lementation fileTree(dir: 'libs', include: ['*.jar'])</pre>                          |
|                  |   | 51 imp                        | lementation rootProject.ext.liteavSdk   |
|                  |   | 52 <b>imp</b>                 | lementation 'com.tencent.liteav:LiteAVSDK Player Prem                                     |
|                  |   | 53                            | lementation project('.deprecatedcommon')  |
| ×                |   | dependencie                   | s{}   |

3.1 Add the LiteAVSDK\_Player\_Premium dependencies to dependencies .





```
dependencies {
    implementation 'com.tencent.liteav:LiteAVSDK_Player_Premium:latest.release'
    // If you want to integrate the basic version of the player
    // implementation 'com.tencent.liteav:LiteAVSDK_Player:latest.release'
    implementation project(':superplayerkit')
    // Third-party library for integration of the on-screen commenting feature of
    implementation 'com.github.ctiao:DanmakuFlameMaster:0.5.3'
}
```



If you need to integrate an older version of the LiteAVSDK\_Player\_Premium SDK, view it in MavenCentral and then integrate it as instructed below:



```
dependencies {
    // Integrate the LiteAVSDK_Player_Premium SDK v10.8.0.29000
    implementation 'com.tencent.liteav:LiteAVSDK_Player_Premium:10.8.0.29000'
    // If you want to integrate the basic version of the player
    // implementation 'com.tencent.liteav:LiteAVSDK_Player:latest.release'
}
```

### 🔗 Tencent Cloud

4. In the defaultConfig of app/build.gradle, specify the CPU architecture to be used by the application (currently, LiteAVSDK supports armeabi, armeabi-v7a, and arm64-v8a, which you can configure as needed).



```
ndk {
abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
}
```

 $5. \ Add \ the \ \ {\tt mavenCentral} \ \ library \ to \ the \ \ {\tt build.gradle} \ \ in \ your \ project \ directory.$


```
repositories {
  mavenCentral()
}
```

```
6. Click
```

## 11

**Sync Now** to sync the SDK. If mavenCentral can be connected to, the SDK will be automatically downloaded and integrated into the project very soon.

1. Download the SDK + demo package for Android here.

2. Import SDK/LiteAVSDK\_Player\_Premium\_XXX.aar ( XXX is the version number) into the libs folder under app and copy the Demo/superplayerkit module to the project.

3. Import superplayerkit into setting.gradle in your project directory.



include ':superplayerkit'

4. Open the build.gradle file of the superplayerkit project and modify the constant values of compileSdkVersion , buildToolsVersion , minSdkVersion , targetSdkVersion , and rootProject.ext.liteavSdk .







```
compileSdkVersion 26
buildToolsVersion "26.0.2"

defaultConfig {
  targetSdkVersion 23
  minSdkVersion 19
}

dependencies {
  implementation(name:'LiteAVSDK_Player_Premium_10.8.0.29000', ext:'aar')
}
```



Import the common module into your project as instructed above and configure it.

Configure repositories



```
repositories {
flatDir {
    dirs '../app/libs'
}
```

5. Add dependencies to app/build.gradle :





```
compile(name:'LiteAVSDK_Player_Premium_10.8.0.29000', ext:'aar')
implementation project(':superplayerkit')
// Third-party library for integration of the on-screen commenting feature of the P
implementation 'com.github.ctiao:DanmakuFlameMaster:0.5.3'
```

6. Add the following to the project's build.gradle :



```
allprojects {
   repositories {
        flatDir {
            dirs 'libs'
        }
   }
}
```

7. In defaultConfig of app/build.gradle, specify the CPU architecture to be used by the application (currently, LiteAVSDK supports armeabi, armeabi-v7a, and arm64-v8a).





```
ndk {
   abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
}
```

8. Click **Sync Now** to sync the SDK.

If you do not want to import the AAR library, you can also integrate LiteAVSDK by importing JAR and SO libraries.

1. Download the SDK + demo package for Android here and decompress it. Find

SDK/LiteAVSDK\_Player\_Premium\_XXX.zip ( XXX is the version number) in the SDK directory. After



decompression, you can get the libs directory, which contains the JAR file and folders of SO files as listed below:



2. Copy the Demo/superplayerkit module to your project and import superplayerkit into setting.gradle in your project directory.





include ':superplayerkit'

3. Copy the libs folder obtained by decompression in step 1 to the superplayerkit project root directory.4. Modify the superplayerkit/build.gradle file:





```
compileSdkVersion 26
buildToolsVersion "26.0.2"
defaultConfig {
  targetSdkVersion 23
  minSdkVersion 19
}
```

Import the common module into your project as instructed above and configure it. Configure sourceSets and add the SO library import code.



```
sourceSets{
   main{
        jniLibs.srcDirs = ['libs']
   }
}
```

Configure repositories , add flatDir , and specify the paths of the local repositories.



```
repositories {
flatDir {
    dirs 'libs'
}
```

5. In defaultConfig of app/build.gradle, specify the CPU architecture to be used by the application (currently, LiteAVSDK supports armeabi, armeabi-v7a, and arm64-v8a).



```
ndk {
   abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
}
```

6. Click **Sync Now** to sync the SDK.

At this point, you have completed integrating the RT-Cube Player for Android.

# Step 3. Configure application permissions



Configure permissions for your application in AndroidManifest.xml . LiteAVSDK needs the following permissions:



```
<!--network permission-->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<!--VOD player floating window permission -->
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<!--storage-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

#### <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

#### Network security configuration allows the app to send HTTP requests

For security reasons, starting from Android P, Google requires that requests from apps use encrypted connections. The player SDK will start a localsever proxy to make HTTP requests. If your app's targetSdkVersion is greater than or equal to 28, you can enable the permission to send HTTP requests to 127.0.0.1 through network security configuration. Otherwise, an "java.io.IOException: Cleartext HTTP traffic to 127.0.0.1 not permitted" error will occur during playback, causing the video to fail to play. The configuration steps are as follows:

1. Create a new res/xml/network\_security\_config.xml file in your project and set the network security configuration.



```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
<domain-config cleartextTrafficPermitted="true">
<domain includeSubdomains="true">127.0.0.1</domain>
</domain-config>
</network-security-config>
```

2. Add the following attributes to the application tag in the AndroidManifest.xml file:



```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
```

```
<application android:networkSecurityConfig="@xml/network_security_config"
... >
...
</application>
</manifest>
```

# Step 4. Set obfuscation rules

In the proguard-rules.pro file, add the classes related to the TRTC SDK to the "do not obfuscate" list:



```
-keep class com.tencent.** { *;}
```

At this point, you have completed configuring permissions for the RT-Cube Player application for Android.

## Step 5. Use the player features

This step describes how to create a player and use it for video playback.

#### 1. Player creation

The main class of the player is SuperPlayerView, and videos can be played back after it is created. FileId or URL can be integrated for playback. Create SuperPlayerView in the layout file:



<!-- Player component --> <com.tencent.liteav.demo.superplayer.SuperPlayerView

# 🕗 Tencent Cloud

android:id="@+id/superVodPlayerView"
android:layout\_width="match\_parent"
android:layout\_height="200dp" />

#### 2. License configuration

If you have obtained a license, you can view the license URL and key in the VOD console.

If you don't have the required license yet, you can get it as instructed in Video Playback License.

After obtaining the License information, you need to initialize and configure the License before calling the relevant interfaces of the SDK. For

detailed tutorials, please see Configuring View License.

#### 3. Video playback

This step describes how to play back a video. The RT-Cube Player for Android can be used for VOD and live playback as follows:

VOD playback: The Player component supports two VOD playback methods, namely, through FileId or URL. Live playback: The Player component can use the playback through URL method for live playback. A live audio/video stream can be pulled for playback simply by passing in its URL. For more information on how to generate a Tencent Cloud live streaming URL, see Splicing Live Streaming URLs.

VOD and live playback through URL

VOD playback through `FileId`

A URL can be the playback address of a VOD file or the pull address of a live stream. A video file can be played back simply by passing in its URL.





```
SuperPlayerModel model = new SuperPlayerModel();
model.appId = 1400329073; // Configure `AppId`
model.url = "http://your_video_url.mp4"; // Configure a URL for your video for pl
mSuperPlayerView.playWithModelNeedLicence(model);
```

A video file ID is returned by the server after the video is uploaded.

- 1. After a video is published from a client, the server will return a file ID to the client.
- 2. After a video is uploaded to the server, the notification for successful upload will contain a file ID for the video.
- If the video you want to play is already saved with VOD, you can go to Media Assets to view its file ID.

|  | Video Name/ID |   | Video Status | Video Cate <b>T</b> | Uploading 🗘            | Expiration Time | 44     |
|--|---------------|---|--------------|---------------------|------------------------|-----------------|--------|
|  | 00:00:10      | test_2022-04-15-17<br>ID:387702299327461625 | ✓ Normal     | Other               | 2022-04-15<br>17:47:24 | Permanent       | (<br>1 |
|  | 00:00:11      | test_2022-04-07-16<br>ID:387702298823695918 | ⊘ Normal     | Other               | 2022-04-07<br>16:14:00 | Permanent       | (<br>r |
|  | 00:00:04      | test_2022-04-07-16<br>ID:387702298823695507 | ⊘ Normal     | Other               | 2022-04-07<br>16:12:07 | Permanent       | (<br>r |

### Note

1. To play by VOD file ID, you need to use the Adaptive-

HLS template (ID: 10) to transcode the video or use the player signature `psign` to specify the video to play; otherwis e, the playback may fail. For more information on how to transcode a video and generate `psign`, see Play back a vid eo with the Player component and Player Signature.

2.

If a "no v4 play info" exception occurs during playback through `FileId`, the above problem may exist. In this case, we recommend you make adjustments as instructed above. You can also directly get the playback link of the source vide o for playback through URL.

3.

We recommend you transcode videos for playback because untranscoded videos may experience compatibility issue s during playback.





```
// If you haven't enabled hotlink protection and a "no v4 play info" error occurs,
SuperPlayerModel model = new SuperPlayerModel();
model.appId = 1400329071;// Configure AppId
model.videoId = new SuperPlayerVideoId();
model.videoId.fileId = "5285890799710173650"; // Configure `FileId`
// `psign` is a player signature. For more information on the signature and how to
model.videoId.pSign = "eyJhbGciOiJIUzI1NiISInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTQwMDMyOTA
mSuperPlayerView.playWithModelNeedLicence(model);
```





## 4. Playback exit

If the player is no longer needed, call resetPlayer to reset the player and free up memory.



```
mSuperPlayerView.resetPlayer();
```

At this point, you have learned how to create a player, use it to play videos, and stop playback.

# More Features

This section describes several common player features. For more features, see Demo. For features supported by the Player component, see Features.

# 1. Full screen playback

The Player component supports full screen playback. In full screen mode, users can lock the screen, control volume and brightness with gestures, send on-screen comments, take screenshots, and switch the video definition. You can try out this feature in TCToolkit App > Player > Player Component, and you can enter the full screen playback mode by clicking the full screen icon in the bottom-right corner.

You can call the API below to enter full screen from the windowed playback mode:





mControllerCallback.onSwitchPlayMode(SuperPlayerDef.PlayerMode.FULLSCREEN);

#### Features of full screen playback mode

Return to the window Screen lock On-screen comments Screenshot Change resolution Click **Back** to return to the window playback mode.





```
// API triggered after tapping
mControllerCallback.onBackPressed(SuperPlayerDef.PlayerMode.FULLSCREEN);
onSwitchPlayMode(SuperPlayerDef.PlayerMode.WINDOW);
```

Screen locking disables touch screen and allows users to enter an immersive playback mode.



```
// API triggered after tapping
toggleLockState();
```

After the on-screen commenting feature is enabled, text comments sent by users will be displayed on the screen.





// Step 1. Add an on-screen comment to the on-screen comment view
addDanmaku(String content, boolean withBorder);
// Step 2. Enable or disable on-screen commenting
toggleBarrage();

The Player component allows users to take and save a screenshot of a video during playback. Click the button in image 4 to capture the screen, and you can save the captured screenshot with the mSuperPlayer.snapshot API.





Users can change the video definition (such as SD, HD, and FHD) during playback.





```
// The API for displaying the definition selection view triggered after the button
showQualityView();
// The callback API for tapping the definition option is as follows
mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    // The event of tapping the definition list view
    VideoQuality quality = mList.get(position);
    mCallback.onQualitySelect(quality);
    }
});
```

```
// Callback for the selected definition
@Override
public void onQualityChange(VideoQuality quality) {
    mFullScreenPlayer.updateVideoQuality(quality);
    mSuperPlayer.switchStream(quality);
}
```

## 2. Floating window playback

The Player component supports playback in a small floating window, which allows users to switch to another application without interrupting the video playback. You can try out this feature in **TCToolkit App** > **Player** > **Player Component** by clicking **Back** in the top-left corner.



Floating window playback relies on the following permission in AndroidManifest :





<uses-permission android:name="android.permission.SYSTEM\_ALERT\_WINDOW" />





// The API triggered by switching to the floating window
mSuperPlayerView.switchPlayMode(SuperPlayerDef.PlayerMode.FLOAT);
// The API triggered by tapping the floating window to return to the main window
mControllerCallback.onSwitchPlayMode(SuperPlayerDef.PlayerMode.WINDOW);

## 3. Thumbnail

The Player component supports customizing a video thumbnail, which is displayed before the callback is received for playing back the first video frame. You can try out this feature in **TCToolkit App** > **Player** > **Player Component** > **Thumbnail Customization Demo**.

When the Player component is set to the automatic playback mode PLAY\_ACTION\_AUTO\_PLAY, the thumbnail will be displayed before the first video frame is loaded.

When the Player component is set to the manual playback mode PLAY\_ACTION\_MANUAL\_PLAY, videos are played only after users tap the play button, and the thumbnail will be displayed until the first video frame is loaded. You can set the thumbnail by specifying the URL of a local or online file. For detailed directions, see the code below. If you play by VOD file ID, you can also set the thumbnail in the VOD console.



```
SuperPlayerModel model = new SuperPlayerModel();
model.appId = "Your `appid`";
model.videoId = new SuperPlayerVideoId();
```

```
model.videoId.fileId = "Your `fileId`";
// Playback mode, which can be set to automatic (`PLAY_ACTION_AUTO_PLAY`) or manual
model.playAction = PLAY_ACTION_MANUAL_PLAY;
// Specify the URL of an online file to use as the thumbnail. If `coverPictureUrl`
model.coverPictureUrl = "http://1500005830.vod2.myqcloud.com/6c9a5118vodcq150000583
mSuperPlayerView.playWithModelNeedLicence(model);
```

## 4. Video playlist loop

The Player component supports looping video playlists.

After a video ends, the next video in the list can be played automatically or users can manually start the next video. After the last video in the list ends, the first video in the list will start automatically.

This feature can be tried out in **TCToolkit App** > **Player** > **Player Component** > **Video List Loop Demo**.


```
// Step 1. Create a loop list<SuperPlayerModel>
ArrayList<SuperPlayerModel> list = new ArrayList<>();
SuperPlayerModel model = new VideoModel();
model = new SuperPlayerModel();
model.videoId = new SuperPlayerVideoId();
model.appid = 1252463788;
model.videoId.fileId = "4564972819219071568";
list.add(model);
model = new SuperPlayerModel();
model.videoId = new SuperPlayerVideoId();
```

```
model.appid = 1252463788;
model.videoId.fileId = "4564972819219071679";
list.add(model);
// Step 2. Call the loop API
mSuperPlayerView.playWithModelListNeedLicence(list, true, 0);
```



public void playWithModelListNeedLicence(List<SuperPlayerModel> models, boolean isL

API parameters:

Parameter Type

Description



| models         | List <superplayermodel></superplayermodel> | Loop data list   |  |
|----------------|--|--|--|
| isLoopPlayList | boolean                                    | Whether to loop video playback                             |  |
| index          | int  | Index of SuperPlayerModel from which to start the playback |  |

#### 5. Picture-in-picture

Starting from Android 8.0 (API level 26), Android allows launching activities in picture-in-picture (PiP) mode.



### 🔗 Tencent Cloud

If you need to enable or disable picture-in-picture, simply change the value of enablePIP in SuperPlayerGlobalConfig. To add picture-in-picture to your app, you need to add the following attribute to the activity that supports picture-inpicture in AndroidManifest.



| <activ< th=""><th>vity&gt;</th></activ<> | vity>  |
|--|--|
| an                                       | ndroid:name=".demo.SuperPlayerActivity"  |
| an                                       | ndroid:resizeableActivity="true"   |
| an                                       | ndroid:supportsPictureInPicture="true"   |
| an                                       | ndroid:documentLaunchMode="intoExisting"                                       |
| an                                       | ndroid:excludeFromRecents="true"   |
| an                                       | ndroid:configChanges="orientation keyboardHidden screenSize smallestScreenSize |



#### </activity>

At the same time, the lifecycle of the activity that supports picture-in-picture needs to be handled specially according to SuperPlayerActivity. To enable picture-in-picture, use PictureInPictureHelper in SuperPlayerView.



```
PictureInPictureHelper mPictureInPictureHelper = new PictureInPictureHelper(mContex
mPictureInPictureHelper.setListener(this);
mPictureInPictureHelper.enterPictureInPictureMode(getPlayerState(),mTXCloudVideoVie
```

You need to release it in SuperPlayerView when exiting.





#### mPictureInPictureHelper.release();

If you need to modify the time interval for moving the custom button forward or backward in picture-in-picture, simply modify the value of PIP\_TIME\_SHIFT\_INTERVAL in PictureInPictureHelper.

#### 6. Preview

The Player component supports the video preview feature, which allows non-member viewers to view a preview of the video. You can pass in different parameters to control the video preview duration, prompt message, and preview end



screen. You can try out this feature in **Tencent Cloud Toolkit App** > **Player** > **Player Component** > **Preview Feature Demo**.



Method 1: // Step 1. Create a video model SuperPlayerModel mode = new SuperPlayerModel(); //... Add the video source information // Step 2. Create a preview information model VipWatchModel vipWatchModel = new VipWatchModel("You can preview %ss and activate mode.vipWatchMode = vipWatchModel; // Step 3. Call the method for playing back videos

```
mSuperPlayerView.playWithModelNeedLicence(mode);
Method 2:
// Step 1. Create a preview information model
VipWatchModel vipWatchModel = new VipWatchModel("You can preview %ss and activate
// Step 2. Call the method for setting the preview feature
mSuperPlayerView.setVipWatchModel(vipWatchModel);
```



public VipWatchModel(String tipStr, long canWatchTime)

VipWatchModel API parameter description:



| Parameter    | Туре   | Description                 |
|--------------|--------|-----------------------------|
| tipStr       | String | Preview prompt message      |
| canWatchTime | Long   | Preview duration in seconds |

#### 7. Dynamic watermark

The Player component allows you to add a randomly moving text watermark to protect your content against piracy. Watermarks are visible in both the full screen mode and windowed mode. The text, font size, and color of a watermark are customizable. You can find a demo for this feature in the TCToolkit app: **Player > Player Component > Dynamic Watermark Demo**.





Method 1: // Step 1. Create a video model SuperPlayerModel mode = new SuperPlayerModel(); //... Add the video source information // Step 2. Create a watermark information model DynamicWaterConfig dynamicWaterConfig = new DynamicWaterConfig("shipinyun", 30, Co mode.dynamicWaterConfig = dynamicWaterConfig; // Step 3. Call the method for playing back videos mSuperPlayerView.playWithModelNeedLicence(mode);

Method 2:



```
// Step 1. Create a watermark information model
DynamicWaterConfig dynamicWaterConfig = new DynamicWaterConfig("shipinyun", 30, Co
// Step 2. Call the method for setting the dynamic watermark feature
mSuperPlayerView.setDynamicWatermarkConfig(dynamicWaterConfig);
```



#### public DynamicWaterConfig(String dynamicWatermarkTip, int tipTextSize, int tipTextC

#### API parameters:

 Parameter
 Type
 Description



| dynamicWatermarkTip | String | Watermark text information |
|---------------------|--------|----------------------------|
| tipTextSize         | int    | Text size                  |
| tipTextColor        | int    | Text color                 |

#### 8. Video download

Video download allows users to cache online videos and watch them offline. The cached video can be played back only in the client but cannot be actually downloaded to the device. This feature can effectively prevent downloaded videos from being distributed without authorization and protect the video security.

You can try out this feature in full screen mode in TCToolkit App > Player > Player Components > Offline Cache.



DownloadMenuListView (cache selection list view) is used to select and download videos at different definitions. After selecting the definition in the top-left corner, click the option of the video to be downloaded. When a check mark appears, the download has started. After clicking the **video download list** button below, you will be redirected to the Activity where VideoDownloadListView is located.





// Step 1. Initialize the download data with the following parameters
DownloadMenuListView mDownloadMenuView = findViewById(R.id.superplayer\_cml\_cache\_me
mDownloadMenuView.initDownloadData(superPlayerModelList, mVideoQualityList, mDefaul

```
// Step 2. Set the options of the video being played back
mDownloadMenuView.setCurrentPlayVideo(mSuperplayerModel);
```

```
// Step 3. Set the click event of the **video download list** button
mDownloadMenuView.setOnCacheListClick(new OnClickListener() {
    @Override
    public void onClick(View v) {
```



```
// Redirect to the Activity where `VideoDownloadListView` is located
    startActivity(DownloadMeduListActivity.this,VideoDownloadListActivity.class)
  }
});
// Step 4. Display the view with animation
```

mDownloadMenuView.show();





String userName)

#### API parameters:

| Parameter            | Туре                                       | Description                  |
|----------------------|--|------------------------------|
| superPlayerModelList | List <superplayermodel></superplayermodel> | The downloaded video data    |
| qualityList          | List <videoquality></videoquality>         | The video definition data    |
| currentQuality       | VideoQuality                               | The current video definition |
| userName             | String                                     | The username                 |

VideoDownloadListView (video download list) displays the list of views of all the videos that are being downloaded and have been downloaded. When this button is clicked, if the download is in progress, it will be paused; if it is paused, it will be resumed; if it has completed, the video will be played back.







// Step 1. Bind the control
VideoDownloadListView = findViewById(R.id.video\_download\_lis

//Step 2. Add data
mVideoDownloadListView.addCacheVideo(mDataList, true);

API parameters:





#### public void addCacheVideo(List<TXVodDownloadMediaInfo> mediaInfoList, boolean isNee

| Parameter     | Туре   | Description                        |
|---------------|--|------------------------------------|
| mediaInfoList | List <txvoddownloadmediainfo></txvoddownloadmediainfo> | The type of the added video data   |
| isNeedClean   | boolean  | Whether to clear the previous data |

### 9. Image sprite and timestamp information



#### Timestamp information

You can add text descriptions at key positions on the progress bar, which the user can click to view and quickly understand the video information at the current position. After clicking the video information, the user can seek to the desired position.

You can try out this feature in full screen mode in TCToolkit App > Player > Player Components > Tencent Cloud Video.



#### Image sprite

Users can view video thumbnails when dragging or seeking on the progress bar so as to quickly understand the video content at the specified position. The thumbnail preview is implemented based on the video's image sprite. You can generate the image sprite of a video file in the VOD console or directly generate an image sprite file. You can try out this feature in full screen mode in TCToolkit App > Player > Player Components > Tencent Cloud Video.









// Step 1. Get the image sprite and timestamp information in the `onPlayEvent` call
mSuperplayerView.play(superplayerModel);

```
// Step 2. Get timestamp and image sprite information in the `VOD_PLAY_EVT_GET_PLAY
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
    switch (event) {
        case TXVodConstants.VOD_PLAY_EVT_GET_PLAYINFO_SUCC:
        // Get the list of image URLs of the image sprite
        playImageSpriteInfo.imageUrls = param.getStringArrayList(TXVodConstants)
```

```
//\ensuremath{\,{\rm Get}} the download URL of the image sprite WebVTT file
```

```
playImageSpriteInfo.webVttUrl = param.getString(TXVodConstants.EVT_IMAG
            // Get the timestamp information
           ArrayList<String> keyFrameContentList =
                    param.getStringArrayList(TXVodConstants.EVT_KEY_FRAME_CONTENT_L
            // Get the time information of the timestamp information
            float[] keyFrameTimeArray = param.getFloatArray(TXVodConstants.EVT_KEY_
            // Construct the list of timestamp information
            if (keyFrameContentList != null && keyFrameTimeArray != null
                    && keyFrameContentList.size() == keyFrameTimeArray.length) {
                for (int i = 0; i < keyFrameContentList.size(); i++) {</pre>
                    PlayKeyFrameDescInfo frameDescInfo = new PlayKeyFrameDescInfo()
                    frameDescInfo.content = keyFrameContentList.get(i);
                    frameDescInfo.time = keyFrameTimeArray[i];
                    mKeyFrameDescInfoList.add(frameDescInfo);
                }
            }
            break;
        default:
            break;
   }
}
// Step 3. Assign the obtained timestamp information and image sprite to the corres
// The view of the image sprite corresponds to `mIvThumbnail` in the `VideoProgress
// The view of the timestamp information corresponds to `TCPointView` in the `Point
updateVideoImageSpriteAndKeyFrame(playImageSpriteInfo,keyFrameDescInfoList);
```

#### 10. External subtitles

#### Note:

External subtitles depend on the premium version SDK of the media player, and the SDK needs to be version 11.3 or above to support it.



Currently, only SRT and VTT subtitle formats are supported. The usage is as follows:

#### Step 1: Add external subtitles.

Pass the external subtitle category field to SuperPlayerModel#subtitleSourceModelList .





```
// Pass in subtitle url, subtitle name, and subtitle type
SubtitleSourceModel subtitleSourceModel = new SubtitleSourceModel();
subtitleSourceModel.name = "ex-cn-srt";
subtitleSourceModel.url = "https://mediacloud-76607.gzc.vod.tencent-cloud.com/DemoR
subtitleSourceModel.mimeType = TXVodConstants.VOD_PLAY_MIMETYPE_TEXT_SRT;
model.subtitleSourceModelList.add(subtitleSourceModel);
```

```
// Play
mSuperPlayerView.playWithModelNeedLicence(model);
```

#### Step 2: Switch subtitles after playback.



```
// After the video starts playing, select the added external subtitles.
public void onClickSubTitleItem(TXTrackInfo clickInfo) {
   List<TXTrackInfo> subtitleTrackInfoList = mVodPlayer.getSubtitleTrackInfo();
   for (TXTrackInfo trackInfo : subtitleTrackInfoList) {
      if (trackInfo.trackIndex == clickInfo.trackIndex) {
         // Select the subtitle
         mVodPlayer.selectTrack(trackInfo.trackIndex);
         mSelectedSubtitleTrackInfo = trackInfo;
      } else {
```



```
// Deselect other subtitles if they are not needed.
mVodPlayer.deselectTrack(trackInfo.trackIndex);
}
}
```

#### Step 3: Configure subtitle styles.

Subtitle styles can be configured before or during playback.



TXSubtitleRenderModel model = new TXSubtitleRenderModel(); model.canvasWidth = 1920; // Subtitle render canvas width



```
model.canvasHeight = 1080; // Subtitle render canvas height
model.fontColor = 0xFFFFFFF; // Set the subtitle font color, default white and opa
model.isBondFontStyle = false; // Set whether the subtitle font is bold
mVodPlayer.setSubtitleStyle(model);
```

#### 111. Ghost watermark

The content of the ghost watermark is filled in the player signature, and is finally displayed on the player through the cloud on-demand backend. The entire transmission link process is coordinated by the cloud and the player to ensure the security of the watermark. Tutorial on configuring ghost watermark in player signature. The ghost watermark only appears on the video for a short period of time, and this flash has little impact on the viewing of the video. The position of the watermark on the screen is not fixed each time, preventing others from trying to block the watermark. The effect is as shown in the picture below. When the video starts playing, the watermark will appear once and then disappear. Wait until the next time it appears before disappearing.

The content of the ghost watermark is obtained through param.getString(TXVodConstants.EVT\_KEY\_WATER\_MARK\_TEXT) after receiving the TXVodConstants#VOD\_PLAY\_EVT\_GET\_PLAYINFO\_SUCC event from the player. Note: Supported from player version 11.6.









```
+ "lFUgBLntfX03iGxK9ntD_AONClUUno";
mSuperPlayerView.playWithModelNeedLicence(model);
// Step 2: After receiving the ghost watermark content callback in SuperPlayerView#
public void onRcvWaterMark(String text, long duration) {
    if (!TextUtils.isEmpty(text)) {
        DynamicWaterConfig dynamicWaterConfig = new DynamicWaterConfig(text, 30, Co
        dynamicWaterConfig.durationInSecond = duration;
        dynamicWaterConfig.setShowType(DynamicWaterConfig.GHOST_RUNNING);
        setDynamicWatermarkConfig(dynamicWaterConfig);
    }
}
```

### Demo

To try out more features, you can directly run the demo project or scan the QR code to download the TCToolkit App demo.

#### Running a demo project

1. Select File > Open on the navigation bar of Android Studio. In the pop-up window, select the

\$SuperPlayer\_Android/Demo directory of the **demo** project. After the demo project is imported successfully,

click Run app to run the demo.

2. After running the demo successfully, go to **Player** > **Player Component** to try out the player features.

# Flutter Integration Guide

Last updated : 2024-04-11 16:11:38

# SDK Download

The Tencent Cloud Player SDK for Flutter can be downloaded here.

### **Intended Audience**

This document describes some of the proprietary capabilities of Tencent Cloud. Make sure that you have activated the relevant Tencent Cloud services before reading this document. If you don't have an account yet, sign up for free trial first.

### This Document Describes

How to integrate the Tencent Cloud Player SDK for Flutter. How to use the Player component for VOD playback.

## Player Component Overview

The Player component for Flutter is an extension of the Player SDK for Flutter. Compared with the VOD player, the Player component is easier to use and integrates more features, including full screen playback, definition selection, progress bar, playback control, and thumbnails. To integrate Flutter video playback capabilities more easily, you can use the Player component for Flutter. To integrate Flutter video playback capabilities more easily, you can use the Superplayer SDK for Flutter. Supported features: Full screen playback Adaptive screen rotation during playback Custom video thumbnail Definition selection Audio and brightness adjustment Playback speed change Hardware acceleration Picture-in-picture (PiP) on Android and iOS Schement Cloud

Image sprite and keyframe timestamp information More features to come soon.

# Integration Guide

- 1. Copy the superplayer\_widget directory from the project to your own Flutter project.
- 2. Add the dependency to your project's configuration file <code>pubspec.yaml</code> .



superplayer\_widget:

```
# The path should be changed according to the location where superplayer_widget
path: ../superplayer_widget
super_player:
git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: main
```

You can replace ref with the corresponding version or branch according to your own project needs.

3. Modify the superPlayer dependency of superplayer\_widget.

Enter the pubspec.yaml file of superplayer\_widget and make the necessary modifications.

Replace the configuration with the following:



super\_player:
 path: ../

Replace with:



```
super_player:
git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: main
```

#### You can replace ref with the corresponding version or branch according to your own project needs.

4. As the player component is now integrated with internationalization, it is necessary to add the internationalization component in the entry function, as shown in the following example:





```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    localizationsDelegates: [
      SuperPlayerWidgetLocals.delegate,
      // ..... your app other delegate
    ],
    supportedLocales: [
      Locale.fromSubtags(languageCode: 'en'),
      Locale.fromSubtags(languageCode: 'zh'),
```

### 🕗 Tencent Cloud

```
// ..... other language
],
// ..... your app other code
);
}
```

5. Import the dependency package of superplayer\_widget on the pages where it is needed, as shown below:



import 'package:superplayer\_widget/demo\_superplayer\_lib.dart';

6. For other native-related configurations, please refer to the Integration Guide.
# **SDK** Integration

## Step 1. Apply for and integrate a video playback license

Before you integrate the player, you need to sign up for a Tencent Cloud account, apply for the video playback license, and configure the license as follows (we recommend you do this when the application is launched): If you don't configure a license, errors may occur during playback.



String licenceURL = ""; // The license URL obtained
String licenceKey = ""; // The license key obtained
SuperPlayerPlugin.setGlobalLicense(licenceURL, licenceKey);

## Step 2. Set the SDK connection environment

In order to help you conduct business with higher quality and security in compliance with applicable laws and regulations in different countries and regions, Tencent Cloud provides two SDK connection environments. If you serve global users, we recommend you use the following API to configure the global connection environment.



SuperPlayerPlugin.setGlobalEnv("GDPR");

## Step 3. Create a controller





SuperPlayerController \_ controller = SuperPlayerController(context);

Step 4. Configure the player





```
FTXVodPlayConfig config = FTXVodPlayConfig();
// If `preferredResolution` is not configured, the 720x1280 resolution stream will
config.preferredResolution = 720 * 1280;
_controller.setPlayConfig(config);
```

For detailed configuration in FTXVodPlayConfig , see the player configuration API of the VOD player SDK for Flutter.

## Step 5. Configure event listening





```
_controller.onSimplePlayerEventBroadcast.listen((event) {
   String evtName = event["event"];
   if (evtName == SuperPlayerViewEvent.onStartFullScreenPlay) {
      setState(() {
        _isFullScreen = true;
      });
   } else if (evtName == SuperPlayerViewEvent.onStopFullScreenPlay) {
      setState(() {
        _isFullScreen = false;
      });
   } else {
   }
}
```

Video on Demand



```
print(evtName);
});
```

## Step 6. Add a layout



```
Widget _getPlayArea() {
    return Container(
    height: 220,
    child: SuperPlayerView(_controller),
```

); }

## Step 7. Listen on the Back button clicking event

Add listening for the return event to ensure that the full screen mode is exited first if the player is in full screen mode when the return event is triggered, and the page will be exited only when the return event is triggered again. If you want to directly exit the page in full screen playback mode, you don't need to implement the listening.



```
Coverride
Widget build(BuildContext context) {
 return WillPopScope(
      child: Container(
        decoration: BoxDecoration(
            image: DecorationImage(
              image: AssetImage("images/ic_new_vod_bg.png"),
              fit: BoxFit.cover,
            )),
        child: Scaffold(
          backgroundColor: Colors.transparent,
          appBar: _isFullScreen
              ? null
              : AppBar(
            backgroundColor: Colors.transparent,
            title: const Text('SuperPlayer'),
          ),
          body: SafeArea(
            child: Builder(
              builder: (context) => getBody(),
            ),
          ),
        ),
      ),
      onWillPop: onWillPop);
}
Future<bool> onWillPop() async {
 return !_controller.onBackPress();
}
```

## Step 8. Start the playback

Through the URL Through `fileId`





```
SuperPlayerModel model = SuperPlayerModel();
model.videoURL = "http://1400329073.vod2.myqcloud.com/d62d88a7vodtranscq1400329073/
_controller.playWithModelNeedLicence(model);
```





```
SuperPlayerModel model = SuperPlayerModel();
model.appId = 1500005830;
model.videoId = new SuperPlayerVideoId();
model.videoId.fileId = "8602268011437356984";
// `psign` is a player signature. For more information on the signature and how to
model.videoId.pSign = "psignXXX"
_controller.playWithModelNeedLicence(model);
```

Find the target video file in Media Assets, and you can view the FileId below the filename.

## 🕗 Tencent Cloud

Play back the video through the FileId , and the player will request the backend for the real playback URL. If the network is abnormal or the FileId doesn't exist, the SuperPlayerViewEvent.onSuperPlayerError event will be received.

## Step 9. Stop the playback

**Remember to call the controller termination method** when stopping the playback, especially before the next call of startVodPlay . This can prevent memory leak and screen flashing issues, as well as ensure that playback is stopped when the page is exited.



Coverride

```
void dispose() {
   // must invoke when page exit.
   _controller.releasePlayer();
   super.dispose();
}
```

# Player Component APIs

## 1. Playing back a video

#### Note

Starting from v10.7, startPlay is replaced by startVodPlay, and playback will succeed only after you use {@link SuperPlayerPlugin#setGlobalLicense} to set the license; otherwise, playback will fail (black screen occurs). The license needs to be set only once globally. You can use the license for CSS, UGSV, or video playback. If you have no such licenses, you can quickly apply for a trial license.

#### Description

This API is used to start video playback.





\_controller.playWithModelNeedLicence(model);

## Parameter description

## 1. SuperPlayerModel

| Parameter | Туре   | Description   |
|-----------|--------|---|
| appld     | int    | The application's appId , which is required for playback via fileId . |
| videoURL  | String | The video URL, which is required for playback via URL.                |



| multiVideoURLs       | List <string></string> | Multi-bitrate playback URLs, which are required for playback via multi-bitrate URLs.   |
|----------------------|------------------------|--|
| defaultPlayIndex     | int                    | The default playback bitrate number, which is used together with <code>multiVideoURLs</code> .                                       |
| videold              | SuperPlayerVideold     | The fileId storage object, which is further described below.   |
| title                | String                 | The video title. You can use this to customize the title and overwrite the title internally requested by the player from the server. |
| coverUrl             | String                 | The thumbnail image pulled from the Tencent server, whose value will be assigned automatically in SuperVodDataLoader .               |
| customeCoverUrl      | String                 | A custom video thumbnail. This parameter is used preferentially and is used to customize the video thumbnail.                        |
| duration             | int                    | The video duration in seconds.   |
| videoDescription     | String                 | The video description.   |
| videoMoreDescription | String                 | The detailed video description.  |
| playAction           | int                    | Valid values: PLAY_ACTION_AUTO_PLAY,<br>PLAY_ACTION_MANUAL_PLAY,<br>PLAY_ACTION_PRELOAD, as described below.                         |

#### 2. SuperPlayerVideoId

| Parameter | Туре   | Description   |
|-----------|--------|---|
| fileId    | String | The file ID, which is required.   |
| psign     | String | The player signature. For more information on the signature and how to generate it, see Player Signature. |

#### 3. playAction

PLAY\_ACTION\_AUTO\_PLAY: The video will be automatically played back after playWithModel is called. PLAY\_ACTION\_MANUAL\_PLAY: The video needs to be played back manually after playWithModel is called. The player doesn't load the video and only displays the thumbnail image, which consumes no video playback resources compared with PLAY\_ACTION\_PRELOAD . PLAY\_ACTION\_PRELOAD: The player will display the thumbnail image and won't start the video playback after playWithModel is called, but the video will be loaded. This can start the playback faster than PLAY\_ACTION\_MANUAL\_PLAY .

## 2. Playback pause

#### Description

This API is used to pause video playback.



\_controller.pause();



## 3. Resuming playback

## Description

This API is used to resume the playback.

API



\_controller.resume();

## 4. Restarting playback



## Description

This API is used to restart the video playback.

## API



\_controller.reStart();

## 5. Resetting the player

## Description

This API is used to reset the player status and stop the video playback.

API



\_controller.resetPlayer();

## 6. Releasing the player

### Description

This API is used to release the player resources and stop the video playback. After it is called, the controller can no longer be reused.





\_controller.releasePlayer();

## 7. Processing the player Back button event

#### Description

This API is used to determine the action to perform when the Back button is clicked in full screen playback mode. If true is returned, the full screen mode is exited, and the Back button clicking event is consumed; if false is returned, the event is unconsumed.



\_controller.onBackPress();

## 8. Switching the definition

## Description

This API is used to switch the definition of the video being played back in real time.





\_controller.switchStream(videoQuality);

#### **Parameter description**

videoQuality can generally be obtained through \_controller.currentQualiyList (definition list) and \_controller.currentQuality (default definition) after the playback starts. The definition selection capabilities have been integrated into the player. You can click the definition in the bottom-right corner to switch the definition in full screen mode.

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |

| index   | int    | The definition number.                    |
|---------|--------|---|
| bitrate | int    | Bitrate for the definition.               |
| width   | int    | The video width for the definition.       |
| height  | int    | The video height for the definition.      |
| name    | String | The definition abbreviation.              |
| title   | String | Displayed definition name.                |
| url     | String | The multi-bitrate URL, which is optional. |

## 9. Adjusting the playback progress (seek)

## Description

This API is used to adjust the current video playback progress.



\_controller.seek(progress);

## Parameter description

| Parameter | Туре   | Description             |
|-----------|--------|-------------------------|
| progress  | double | Target time in seconds. |

## 10. Configuring the Player component

## Description



This API is used to configure Superplayer.

## API



\_controller.setPlayConfig(config);

## Parameter description

| Parameter         | Туре | Description  |
|-------------------|------|--|
| connectRetryCount | int  | The number of player reconnections. If the SDK is disconnected from the server due to an exception, the SDK will attempt to reconnect to the server. |



| connectRetryInterval     | int    | The interval between two player reconnections. If the SDK is disconnected from the server due to an exception, the SDK will attempt to reconnect to the server.  |
|--------------------------|--------|--|
| timeout                  | int    | Player connection timeout period   |
| playerType               | int    | Player type. Valid values: 0: VOD; 1: live streaming; 2: live stream replay.   |
| headers                  | Мар    | Custom HTTP headers  |
| enableAccurateSeek       | bool   | Whether to enable accurate seek. Default value: true.  |
| autoRotate               | bool   | If it is set to true, the MP4 file will be automatically rotated according to the rotation angle set in the file, which can be obtained from the PLAY_EVT_CHANGE_ROTATION event. Default value: true.                                |
| smoothSwitchBitrate      | bool   | Whether to enable smooth multi-bitrate HLS stream switching. If it is set<br>to false (default), multi-bitrate URLs are opened faster. If it is set to<br>true, the bitrate can be switched smoothly when IDR frames are<br>aligned. |
| cacheMp4ExtName          | String | The cached MP4 filename extension. Default value: mp4 .  |
| progressInterval         | int    | Progress callback interval in ms. If it is not set, the SDK will call back the progress once every 0.5 seconds.  |
| maxBufferSize            | int    | The maximum size of playback buffer in MB. The setting will affect playableDuration. The greater the value, the more the data that is buffered in advance.   |
| maxPreloadSize           | int    | Maximum preload buffer size in MB  |
| firstStartPlayBufferTime | int    | Duration of the video data that needs to be loaded during the first buffering in ms. Default value: 100 ms   |
| nextStartPlayBufferTime  | int    | The minimum buffered data size to stop buffering (secondary buffering for insufficient buffered data or progress bar drag buffering caused by seek) in milliseconds. Default value: 250 ms   |
| overlayKey               | String | The HLS security enhancement encryption and decryption key.  |
| overlaylv                | String | The HLS security enhancement encryption and decryption IV.   |
| extInfoMap               | Мар    | Some special configuration items.  |
| enableRenderProcess      | bool   | Whether to allow the postrendering and postproduction feature, which is<br>enabled by default. If the super-resolution plugin exists after it is enabled,  |



|                     |     | the plugin will be loaded by default.                 |  |  |
|---------------------|-----|---|--|--|
| preferredResolution | int | Resolution of the video used for playback preferably. |  |  |
|                     |     | <pre>preferredResolution = width * height</pre>       |  |  |
|                     |     |   |  |  |

## 11. Enabling/Disabling hardware decoding

## Description

This API is used to enable/disable playback based on hardware decoding.



\_controller.enableHardwareDecode(enable);

## 12. Getting the playback status

#### Description

This API is used to get the playback status.

API



SuperPlayerState superPlayerState = \_controller.getPlayerState();

## Parameter description

| Parameter | Туре             | Description    |
|-----------|------------------|----------------|
| INIT      | SuperPlayerState | Initial status |
| PLAYING   | SuperPlayerState | Playing back   |
| PAUSE     | SuperPlayerState | Paused         |
| LOADING   | SuperPlayerState | Loading        |
| END       | SuperPlayerState | Ended          |

## **13. Entering the PiP mode**

## Description

This API is used to display the current video in PiP mode. The PiP mode can be enabled only on supported device models on Android 7.0 or later.



```
_controller.enterPictureInPictureMode(
backIcon: "images/ic_pip_play_replay.png",
playIcon: "images/ic_pip_play_normal.png",
pauseIcon: "images/ic_pip_play_pause.png",
forwardIcon: "images/ic_pip_play_forward.png");
```

#### **Parameter description**

The parameters are applicable only to Android.

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |



| backlcon    | String | The seek backward icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used. |
|-------------|--------|--|
| playIcon    | String | The playback icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used.      |
| pauselcon   | String | The pause icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used.         |
| forwardIcon | String | The fast forward icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used.  |

# **Event Notifications**

## 1. Listening on playback events

## Description

This callback is used to listen for player operation events.

Code





```
_controller.onSimplePlayerEventBroadcast.listen((event) {
   String evtName = event["event"];
   if (evtName == SuperPlayerViewEvent.onStartFullScreenPlay) {
      setState(() {
      _ isFullScreen = true;
      });
   } else if (evtName == SuperPlayerViewEvent.onStopFullScreenPlay) {
      setState(() {
      _isFullScreen = false;
      });
   } else {
```

```
print(evtName);
}
});
```

#### **Event description**

| Status                  | Description                           |
|-------------------------|---------------------------------------|
| onStartFullScreenPlay   | Entered the full screen playback mode |
| onStopFullScreenPlay    | Exited the full screen playback mode  |
| onSuperPlayerDidStart   | Playback started                      |
| onSuperPlayerDidEnd     | Playback ended                        |
| onSuperPlayerError      | Playback error                        |
| onSuperPlayerBackAction | Return event                          |

# **Advanced Features**

## 1. Requesting video data in advance through fileId

The SuperVodDataLoader API can be used to request the video data in advance to accelerate the playback start process.

Sample code





```
SuperPlayerModel model = SuperPlayerModel();
model.appId = 1500005830;
model.videoId = new SuperPlayerVideoId();
model.videoId.fileId = "8602268011437356984";
model.title = "VOD";
SuperVodDataLoader loader = SuperVodDataLoader();
// Values of the required parameters in `model` are directly assigned in `SuperVodD
loader.getVideoData(model, (resultModel) {
   __controller.playWithModelNeedLicence(resultModel);
})
```

## 2. Using the PiP mode

1. Platform configuration.

Android

IOS

In your project's Android package, find the build.gradle file and make sure that the compileSdkVersion and targetSdkVersion are version 31 or higher.

In your project's target, select Signing & Capabilities and add Background Modes, then check "Audio, AirPlay, and Picture in Picture".

2. Copy the sample code of SuperPlayer .

In example/lib in the GitHub project, copy the SuperPlayer package to the lib directory in your project and integrate the Player component as instructed in demo\_superplayer.dart in the sample code.

Then, you can see the PiP mode button at the center on the right of the playback UI of the Player component and click the button to enter the PiP mode.

3. Listening on the lifecycle of the PiP mode.

You can use onExtraEventBroadcast in SuperPlayerPlugin to listen on the lifecycle of the PiP mode. As follows:





```
SuperPlayerPlugin.instance.onExtraEventBroadcast.listen((event) {
    int eventCode = event["event"];
    if (eventCode == TXVodPlayEvent.EVENT_PIP_MODE_ALREADY_EXIT) {
        // exit pip mode
    } else if (eventCode == TXVodPlayEvent.EVENT_PIP_MODE_REQUEST_START) {
        // enter pip mode
    } else if (eventCode == TXVodPlayEvent.EVENT_PIP_MODE_ALREADY_ENTER) {
        // already enter pip mode
    } else if (eventCode == TXVodPlayEvent.EVENT_IOS_PIP_MODE_WILL_EXIT) {
        // will exit pip mode
    } else if (eventCode == TXVodPlayEvent.EVENT_IOS_PIP_MODE_RESTORE_UI) {
    }
}
```

```
// restore UI only support iOS
}
);
```

4. Error codes for entering the PiP mode

If the user fails to enter the PiP mode, the failure will not only be logged but also be prompted through a toast. You can modify the error handling operations in the \_\_onEnterPipMode method in superplayer\_widget.dart . The error codes are as detailed below:

| Parameter                        | Code | Description   |
|----------------------------------|------|---|
| NO_ERROR                         | 0    | Started successfully with no errors.  |
| ERROR_PIP_LOWER_VERSION          | -101 | The Android version is too early and doesn't support the PiP mode.                                |
| ERROR_PIP_DENIED_PERMISSION      | -102 | The PiP mode permission wasn't enabled, or the current device doesn't support PiP.                |
| ERROR_PIP_ACTIVITY_DESTROYED     | -103 | The current UI was terminated.  |
| ERROR_IOS_PIP_DEVICE_NOT_SUPPORT | -104 | The device model or system version doesn't support PiP (only supported on iPadOS 9+ and iOS 14+). |
| ERROR_IOS_PIP_PLAYER_NOT_SUPPORT | -105 | The player doesn't support PiP.   |
| ERROR_IOS_PIP_VIDEO_NOT_SUPPORT  | -106 | The video doesn't support PiP.  |
| ERROR_IOS_PIP_IS_NOT_POSSIBLE    | -107 | The PiP controller was unavailable.   |
| ERROR_IOS_PIP_FROM_SYSTEM        | -108 | The PiP controller reported an error.   |
| ERROR_IOS_PIP_PLAYER_NOT_EXIST   | -109 | The player object didn't exist.   |
| ERROR_IOS_PIP_IS_RUNNING         | -110 | The PiP feature was running.  |
| ERROR_IOS_PIP_NOT_RUNNING        | -111 | The PiP feature didn't start.   |

5. Checking whether the current device supports PiP

You can use isDeviceSupportPip in SuperPlayerPlugin to check whether the PiP mode can be enabled as follows:




```
int result = await SuperPlayerPlugin.isDeviceSupportPip();
if(result == TXVodPlayEvent.NO_ERROR) {
    // pip support
}
```

The returned result in result means the same as the error code of the PiP mode.

6. Use picture-in-picture controllers to manage picture-in-picture

The picture-in-picture controller TXPipController is a tool for picture-in-picture encapsulated in the

superplayer\_widget, and must be used in conjunction with SuperPlayerView.



When entering picture-in-picture, the current interface will be automatically closed and the listener method set in advance will be called. In the callback method, you can save the necessary parameters of the current interface. After restoring from picture-in-picture, the previous interface will be pushed back and the previously saved parameters will be passed.

When using this controller, only one instance of picture-in-picture and the player can exist. When re-entering the player interface, picture-in-picture will be automatically closed.

6.1 At the entry point of your project, such as main.dart, call TXPipController to set up picture-in-picture control and jump to the player page for entering picture-in-picture.

You can set different interfaces according to your project. The code example is as follows:



# 🕗 Tencent Cloud

```
TXPipController.instance.setNavigatorHandle((params) {
    navigatorKey.currentState?.push(MaterialPageRoute(builder: (_) => DemoSuperPlayer
});
```

6.2 To set up the listener for the picture-in-picture playback page, you need to implement the

TXPipPlayerRestorePagemethod. After setting it up, when you are about to enter picture-in-picture, thecontroller will call thevoid onNeedSavePipPageState(Map<String, dynamic> params)method. Atthis time, you can store the necessary parameters for the current page in the params.method. at



TXPipController.instance.setPipPlayerPage(this);

Later, when the user clicks on the enter picture-in-picture button on the SuperPlayerView, the \_\_onEnterPipMode internal method of SuperPlayerView will be called to enter picture-in-picture, or you can call the enterPictureInPictureMode method of SuperPlayerController to enter it manually.

# 3、Video download

### donwload Video

1. To use the video download feature of the player component, you first need to enable <code>isEnableDownload</code> in SuperPlayerModel, which is disabled by default.



# 🕗 Tencent Cloud

```
SuperPlayerModel model = SuperPlayerModel();
// Enable video download capability
model.isEnableDownload = true;
```

The player component currently only enables downloading in VOD playback mode.

2. You can use the startDownload method of SuperPlayerController to directly download the video that the player is currently playing, corresponding to the clarity of the currently playing video. You can also use DownloadHelper to download a specified video, as follows:



DownloadHelper.instance.startDownloadBySize(videoModel, videoWidth, videoHeight);



You can use startDownloadBySize of DownloadHelper to download videos of a specific resolution. If the resolution is not available, a video with a similar resolution will be downloaded.

In addition to the above interfaces, you can also choose to pass in the quality ID or medialnfo to download directly.



| // QUALITY_2 | 240P | 240p |
|--------------|------|------|
|--------------|------|------|

- // QUALITY\_360P 360P
- // QUALITY\_480P 480p
- // QUALITY\_540P 540p
- // QUALITY\_720P 720p
- // QUALITY\_1080P 1080p

// QUALITY\_2K 2k // QUALITY\_4K 4k // The quality parameter can be customized to take the minimum value of the resolut // (for example, for a resolution of 1280\*720, if you want to download a stream of // you can pass in QUALITY\_720P for the quality parameter). The player SDK will sel // a resolution less than or equal to the passed-in resolution for downloading. // Using quality ID to download DownloadHelper.instance.startDownload(videoModel, qualityId); // Using mediaInfo to download DownloadHelper.instance.startDownloadOrg(mediaInfo);

#### 3. Quality ID conversion

For VOD, CommonUtils provides the getDownloadQualityBySize method to convert the resolution to the corresponding quality ID.





CommonUtils.getDownloadQualityBySize(width, height);

#### Stop downloading video

You can use stopDownload of DownloadHelper method to stop downloading the corresponding video. Example code:





DownloadHelper.instance.stopDownload(mediaInfo);

The medialnfo can be obtained throughgetMediaInfoByCurrentofDownloadHelpermethod or by usinggetDownloadListofTXVodDownloadControllerto obtain download information.

#### Delete downloaded video

You can use deleteDownload of DownloadHelper method to delete the corresponding video





bool deleteResult = await DownloadHelper.instance.deleteDownload(downloadModel.medi

The deleteDownload method will return the result of the deletion, so you can determine whether the deletion was successful.

#### **Download status**

DownloadHelper provides the basic isDownloaded method to determine whether a video has been downloaded. You can also register a listener to determine the download status in real time.

DownloadHelper distributes download events, and you can register for events using the following code.





```
// Registering download event listeners
DownloadHelper.instance.addDownloadListener(FTXDownloadListener((event, info) {
    // Download status change
    }, (errorCode, errorMsg, info) {
        // Download error callback
    }));
// Removing download event listeners
DownloadHelper.instance.removeDownloadListener(listener);
```



In addition, you can also use the TXVodDownloadController.instance.getDownloadInfo(mediaInfo) method or the TXVodDownloadController.instance.getDownloadList() method to directly query the downloadState in the mediaInfo to determine the download status.

#### Play downloaded videos

You can use the playPath field in the video information obtained

fromTXVodDownloadController.instance.getDownloadInfo(mediaInfo)andTXVodDownloadController.instance.getDownloadList()to play the downloaded videos directlywith TXVodPlayerController.



controller.startVodPlay(mediaInfo.playPath);

### 4. The usage of screen orientation

### Screen orientation switching configuration

To enable screen orientation switching for the player component, you need to open the project configuration in Xcode on iOS. Under the "Deployment" tab in the "General" section, select " Landscape left " and " Landscape right ". This will ensure that the iOS device can support landscape orientation.

If you want other pages of your app to remain in portrait mode and not be affected by automatic screen rotation, you need to configure the portrait mode at the entry point of your project. The code is as follows:





SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);

### Automatically switch to full-screen mode according to the sensor configuration

On the Android side, you need to call the following method to start listening to the sensor:





SuperPlayerPlugin.startVideoOrientationService();

After calling this method, the Android device will start listening to the sensor, and rotate events will be sent to the Flutter side via SuperPlayerPlugin.instance.onEventBroadcast . The player component will automatically rotate according to these events. Here is an example of how to use this listener:





```
SuperPlayerPlugin.instance.onExtraEventBroadcast.listen((event) {
    int eventCode = event["event"];
    if (eventCode == TXVodPlayEvent.EVENT_ORIENTATION_CHANGED ) {
        int orientation = event[TXVodPlayEvent.EXTRA_NAME_ORIENTATION];
        // do orientation
    }
});
```

# Demo experience

For more features and a demo experience of debugging, please click here. When running this demo, you need to set your own player license in the demo\_config, and modify the package name and bundleld to your signed package name and bundleld in the Android and iOS configurations.

# Integration (No UI) Web Integration TCPlayer Integration Guide

Last updated : 2024-04-11 16:48:34

This document introduces the Web Player SDK (TCPlayer) tailored for both VOD and live streaming. It can be quickly integrated with your own web application to enable video playback features. TCPlayer comes with a default set of UI elements, which you can use as needed.

# Overview

The web player utilizes the <video> tag of HTML5 and Flash for video playback. It offers a uniform video playback experience across different platforms when browsers do not natively support video playback. In conjunction with Tencent's Video on Demand service, it provides hotlink protection and features for playing standard encrypted HLS videos.

| Audio/Video<br>Protocol | Use             | URL Format                                  | PC<br>Browser | Mobile<br>Browser   |
|-------------------------|-----------------|---|---------------|---------------------|
| MP3                     | Audio           | http://xxx.vod.myqcloud.com/xxx.mp3         | Supported     | Supported           |
| MP4                     | VOD<br>playback | http://xxx.vod.myqcloud.com/xxx.mp4         | Supported     | Supported           |
| HLS(M3U8)               | Live<br>stream  | http://xxx.liveplay.myqcloud.com/xxx.m3u8   | Supported     | Supported           |
|                         | VOD<br>playback | http://xxx.vod.myqcloud.com/xxx.m3u8        | Supported     | Supported           |
| FLV                     | Live<br>stream  | http://xxx.liveplay.myqcloud.com/xxx.flv    | Supported     | Partially supported |
|                         | VOD<br>playback | http://xxx.vod.myqcloud.com/xxx.flv         | Supported     | Partially supported |
| WebRTC                  | Live<br>stream  | webrtc://xxx.liveplay.myqcloud.com/live/xxx | Supported     | Supported           |

# Supported protocols



### Note:

Only H.264 encoding is supported.

The player is compatible with mainstream browsers and can automatically select the optimal playback scheme depending on the browser.

In some browser environments, HLS and FLV video playback depends on Media Source Extensions.

If a browser does not support WebRTC, a WebRTC URL passed in will be converted automatically to better support playback.

# Supported Features

| Feature\\Browser                  | Chrome | Firefox | Edge     | QQ<br>Browser | Mac<br>Safari | iOS<br>Safari | WeChat | Android<br>Chrome |
|-----------------------------------|--------|---------|----------|---------------|---------------|---------------|--------|-------------------|
| Player dimension configuration    | 1      | 1       | 1        | 1             | 1             | 1             | 1      | 1                 |
| Resuming<br>playback              | 1      | 1       | 1        | <b>√</b>      | 1             | 1             | ✓      | 1                 |
| Playback speed change             | 1      | 1       | 1        | <b>√</b>      | 1             | 1             | ✓      | 1                 |
| Preview<br>thumbnails             | 1      | 1       | 1        | <b>√</b>      | -             | -             | -      | -                 |
| Changing `fileID`<br>for playback | 1      | 1       | 1        | 1             | 1             | 1             | 1      | 1                 |
| Flipping videos                   | 1      | 1       | 1        | 1             | 1             | 1             | 1      | 1                 |
| Progress bar<br>marking           | 1      | 1       | 1        | 1             | 1             | -             | -      | -                 |
| HLS adaptive bitrate              | 1      | 1       | 1        | ✓             | 1             | 1             | 1      | 1                 |
| Referer hotlink protection        | 1      | 1       | 1        | <b>√</b>      | 1             | 1             | 1      | -                 |
| Definition change notifications   | 1      | 1       | <i>√</i> | 1             | -             | -             | -      | 1                 |
| Preview                           | 1      | 1       | 1        | 1             | 1             | 1             | 1      | 1                 |
|                                   |        |         |          |               |               |               |        |                   |

S Tencent Cloud

| Playing HLS<br>videos encrypted<br>using standard<br>schemes  | ~        | <i>✓</i> | J  | 1            | 1 | 1 | ✓<br>               | ✓  |
|---|----------|----------|----|--------------|---|---|---------------------|----|
| Playing HLS<br>videos encrypted<br>using private<br>protocols | 1        | <i>J</i> | 1  | -            | - | - | Android:√<br>iOS: - | 1  |
| Video statistics  | 1        | 1        | 1  | 1            | - | - | -                   | -  |
| Video data<br>monitoring                                      | <i>√</i> | 1        | 1  | <i>✓</i>     | - | - | -                   | -  |
| Custom UI<br>messages   | <i>√</i> | 1        | 1  | <i>✓</i>     | 1 | 1 | ✓                   | \$ |
| Custom UI   | 1        | 1        | 1  | 1            | 1 | 1 | 1                   | 1  |
| On-screen comments  | <i>√</i> | 1        | 1  | <i>✓</i>     | 1 | 1 | ✓                   | \$ |
| Watermark   | 1        | 1        | 1  | $\checkmark$ | 1 | 1 | 1                   | 1  |
| Ghost watermark   | 1        | 1        | 1  | 1            | 1 | 1 | 1                   | 1  |
| Playlist  | 1        | 1        | 1  | 1            | 1 | 1 | 1                   | 1  |
| Frame sync<br>under poor<br>network<br>conditions             | 1        | 1        | s. | 1            | 1 | 1 | 1                   | ✓  |

#### Note:

Only H.264 encoding is supported.

Chrome and Firefox for Windows and macOS are supported.

Chrome, Firefox, Edge, and QQ Browser need to load hls.js to play HLS.

The Referer hotlink protection feature is based on the Referer field of HTTP request headers. Some HTTP requests initiated by Android browsers do not carry the Referer field.

The player is compatible with mainstream browsers and can automatically select the optimal playback scheme depending on the browser used. For example, for modern browsers such as Chrome, the player uses the HTML5 technology for playback, and for mobile browsers, it uses the HTML5 technology or the browser's built-in capabilities.



# Preparations

From version 5.0.0, the TCPlayer SDK for Web (TCPlayer) requires access to a License authorization for use. If you don't need the new premium functions, you can apply for a basic License to **try TCPlayer for free;** if you want to use the newly added premium functions you need to purchase a premium License. The detailed information is as follows:

| TCPlayer<br>feature  | Feature Scope   | Required License                      | Pricing                          | Authorization Unit   |
|----------------------|---|---------------------------------------|----------------------------------|--|
| Basic<br>Functions   | Includes all features<br>provided in versions<br>prior to 5.0.0, see<br>Product Features for<br>details | Web Player Basic<br>Version License   | <b>0 CNY</b> Free<br>Application | An exact domain (1<br>License can<br>authorize up to 10<br>exact domains)  |
| Premium<br>Functions | Basic Version<br>Features, VR<br>Playback, Security<br>Check  | Web Player Premium<br>Version License | 399<br>CNY/month<br>Buy Now      | Wildcard Domain (1<br>License can<br>authorize up to 1<br>wildcard domain) |

### Note:

1. Web Player Basic Version License can be applied for free, with a default validity of 1 year; if the remaining validity period is less than 30 days, it can be renewed for free.

2. To facilitate local development, the player won't authenticate localhost or 127.0.01; hence, these types of local service domain names need not be applied for when requesting a License.

# Integration Guide

By following these steps, you can add a video player to your website.

## Step 1. Import files into the page

The Player SDK supports two integration methods: CDN and NPM:

### 1. Integration through CDN

Create a new index.html file in your local project and import the player style and script files into the HTML page:





```
<link href="https://web.sdk.qcloud.com/player/tcplayer/release/v5.1.0/tcplayer.min
<!--Player script file-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v5.1.0/tcplayer.v5</pre>
```

It is recommended to deploy resources on your own when using the Player SDK, click Download player resources. Deploy the unzipped folder without altering its directory structure to prevent cross-referencing issues between resources.

If the deployment address is aaa.xxx.ccc , import the player style and script files at the appropriate places. When



deploying on your own, you need to manually reference the dependency files under the libs folder of the resource package, otherwise, the Tencent Cloud CDN files will be requested by default.



```
<link href="aaa.xxx.ccc/tcplayer.min.css" rel="stylesheet"/>
<!--To play HLS format videos in modern browsers like Chrome and Firefox through H
<script src="aaa.xxx.ccc/libs/hls.min.x.xx.m.js"></script>
<!--Player script file-->
<script src="aaa.xxx.ccc/tcplayer.vx.x.x.min.js"></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script
```

### 2. Integration through npm



First, install the tcplayer npm package:



npm install tcplayer.js

Import the SDK and style files:





```
import TCPlayer from 'tcplayer.js';
import 'tcplayer.js/dist/tcplayer.min.css';
```

# Step 2. Place the player container

Add the player container to the location on the page where the player is to be displayed. For example, add the following code in index.html (the container ID and dimensions can be custom defined).





<video id="player-container-id" width="414" height="270" preload="auto" playsinline </video>

### Note:

The player container must be a <video> tag.

In the example, player-container-id is the ID of the player container, which you can set yourself.

We recommend you set the size of the player container zone through CSS, which is more flexible than the attribute and can achieve effects such as fit to full screen and container adaption.

The preload attribute in the example specifies whether to load the video after the page is loaded. It is usually set to auto for faster video playback. Other options are: meta (to only load metadata after the page loads) and none (to not load the video after the page loads). Videos will not automatically load on mobile devices due to system restrictions.

The attributes playsinline and webkit-playsinline are used to achieve inline playback in standard mobile browsers without hijacking video playback. This is just an example, please use as needed. Setting the x5-playsinline attribute in the TBS kernel will utilize the X5 UI player.

# Step 3. Initialize the player

After page initialization, you can play video resources. The player supports both video on demand (VOD) and live streaming playback scenarios as follows:

VOD playback: The player can play Tencent Video on Demand media resources through FileID. For the specific VOD process, please refer to the Using the Player for Playback document.

Live playback: The player can pull a live audio/video stream for playback by passing in a URL. For information on generating a Tencent Cloud Streaming Services URL, see Splicing Live Streaming URLs.

URL playback (VOD and live)

File ID playback (VOD)

After page initialization, call the method in the player instance to pass in the URL to the method.



```
// `player-container-id` is the player container ID, which must be the same as in H
var player = TCPlayer('player-container-id', {
    sources: [{
        src: 'Please replace with your playback URL',
        }],
        licenseUrl: 'Please replace with your licenseUrl', // License URL, see the prep
        language: 'Please replace with your Setting language', // Setting language en |
    });
// player.src(url); // URL playback address
```

# ठ Tencent Cloud

In the initialization code on the index.html page, add the following initialization script. Pass in the video ID and appID obtained during the preparation phase from the fileID (in Media Management) and (peek in Account Information > Basic Information).



var player = TCPlayer('player-container-id', { // player-container-id is the player fileID: 'Please enter your fileID', // Enter the fileID of the video to be play appID: 'Please enter your appID', // Enter the appID of your VOD account // Enter the player Signature psign, for information on the Signature and how t psign: 'Please enter your player Signature psign', licenseUrl: 'Please enter your licenseUrl', // Refer to the preparation section language: 'Please replace with your Setting language', // Setting language en | });

## Note:

Not all videos can be played successfully in a browser. We recommend you use Tencent Cloud's services to transcode a video before playing it.

## Step 4. Implement more features

You can utilize the server-side capabilities of Video on Demand (VOD) for advanced features, such as automatic switching of adaptive streams, video thumbnail previews, and adding video marker information. These features are detailed in Play back a long video, which you can refer to for implementation.

Additionally, the player offers more features. For a list of features and instructions on how to use them, please see the Feature Demonstration page.

# **TCPlayer Resolution Configuration Guide**

Last updated : 2024-04-11 16:50:03

During playback, you can switch the video resolution automatically or manually to accommodate different sizes of playback devices and network conditions, thus enhancing the viewing experience. This article will explain several scenarios.

# Live Streaming

Live streaming videos are played in the form of URLs. When initializing the player, you can specify the URL to be played through the sources field. Alternatively, after initializing the player, you can play by calling the src method on the player instance.

# 1. Adaptive Bitrate (ABR)

Adaptive bitrate URLs can seamlessly transition during switches without causing interruptions or jumps, ensuring a smooth transition in both visual and auditory experiences. This technology is also relatively simple to use; you just need to pass the playback URL to the player, which will automatically parse the sub-streams and render the resolution switching component on the control bar.

### Example 1: Playing HLS adaptive bitrate URLs

During the player's initialization, when an adaptive bitrate URL is passed in, the player will automatically generate a resolution switching component and switch automatically based on network conditions.





```
const player = TCPlayer('player-container-id', { // player-container-id is the play
sources: [{
    src: 'https://hls-abr-url', // hls adaptive bitrate URL
    }],
});
```

### Note:

Parsing the substreams of an HLS adaptive bitrate requires the dependency on the MSE API of the playback environment. In browsers not supporting MSE (e.g., Safari on iOS), the browser internally handles this by

automatically switching resolutions based on network conditions, but it won't be able to parse multiple resolutions for manual switching.

### Example 2: Playing WebRTC adaptive bitrate URLs

In the WebRTC adaptive bitrate scenario, after the address is input, the player will automatically decompose the substream addresses based on the ABR template in the address.



```
const player = TCPlayer('player-container-id',{
  sources: [{
    src: 'webrtc://global-lebtest-play.myqcloud.com/live/lebtest?txSecret=f22a813b2
```

```
}],
webrtcConfig: {
    // Whether to render multiple resolutions switch, enabled by default, optional
    enableAbr: true,
    // The label name corresponding to the template name, optional
    abrLabels: {
        d1080p: 'FHD',
        d540p: 'HD',
        d360p: 'SD',
        auto: 'AUTO',
     },
});
```

The following explanations are provided for the parameters in the WebRTC URL:

1. tabr\_bitrates specifies the ABR template, and the number of templates will determine the number of rendered resolutions. If no separate resolution label is set, the template name (e.g., d1080p) will be used as the name of the resolution.

2. tabr\_start\_bitrate specifies the initial resolution setting.

3. tabr\_control sets whether automatic resolution switching is enabled. Once enabled, the player will render an option for automatic resolution.

# 2. Manually setting the resolution

If the playback URL is not an adaptive bitrate URL, you can also manually set the resolution. See the following code:





```
const player = TCPlayer('player-container-id', { // player-container-id is the play
multiResolution:{
    // Configure multiple resolution URLs
    sources:{
        'SD':[{
            src: 'http://video-sd-url',
        }],
        'HD':[{
            src: 'http://video-hd-url',
        }],
        'FHD':[{
            Src: 'http://video-hd-url',
        }],
            Source:[{
            Src: 'http://video-hd-url',
            Source:[{
            Src: 'http://video-hd-url',
            Source:[{
            Src: 'http://video-hd-url',
            Source:[{
            Src: 'http://video-hd-url',
            Source:[{
            Src:[{
            Source:[{
            Src:[{
            Sr
```

```
src: 'http://video-fhd-url',
}]
},
// Configure the tag for each resolution
labels:{
    'SD':'Standard Definition', 'HD':'High Definition', 'FHD':'Full High Definition
    },
    // Configure the order of resolutions in the player component
    showOrder:['SD', 'HD', 'FHD'],
    // Configure the default selected resolution
    defaultRes: 'SD',
},
```

# **VOD** Scenario

In the VOD scenario, if you play via fileID, which type of file to play (source file, transcoded file, adaptive bitrate file) and the resolution of substreams of the adaptive bitrate file are all set in the player signature. You can refer to the guide Play back an adaptive bitrate streaming video to understand the entire process of playing videos in the VOD scenario.

When calculating the player signature, you can set the display names of substreams of different resolutions through the resolutionNames in the contentInfo field. If you leave it blank or fill in an empty array, the default configuration is used.


```
resolutionNames: [{
    MinEdgeLength: 240,
    Name: '240P',
}, {
    MinEdgeLength: 480,
    Name: '480P',
}, {
    MinEdgeLength: 720,
    Name: '720P',
}, {
    MinEdgeLength: 1080,
```

```
Name: '1080P',
}, {
MinEdgeLength: 1440,
Name: '2K',
}, {
MinEdgeLength: 2160,
Name: '4K',
}, {
MinEdgeLength: 4320,
Name: '8K',
}]
```

The number of substreams during playback depends on the number of substreams converted according to different adaptive bitrate templates during transcoding. These substreams will fall within the MinEdgeLength range set by resolutionNames based on short side length and then be displayed with the corresponding Name as the clarity name. If you need to quickly experience generating the player signature, you can use the Tencent Video on Demand console's Player Signature Generation Tool.

## TCPlayer Swift Live Streaming Downgrade Notice

Last updated : 2024-04-11 16:50:14

## Downgrade scenarios

Live Event Broadcasting is based on WebRTC and depends on the operating system and browser support for WebRTC.

Currently, the SDK has been tested on the following operating systems and browsers, with the test results as follows:

| Operating system | OS Version | Browser            | Browser version | Support for stream pull |
|------------------|------------|--------------------|-----------------|-------------------------|
|                  | win 10     | Chrome             | 86+             | ✓                       |
| Windows          |            | Firefox            | 88+             | ✓                       |
|                  |            | Microsoft Edge     | 86+             | $\checkmark$            |
|                  | 10.5+      | Safari             | 13.1+           | ✓                       |
| macOS            |            | Chrome             | 86+             | $\checkmark$            |
| macOS            |            | Firefox            | 88+             | ✓                       |
|                  |            | Microsoft Edge     | 86+             | $\checkmark$            |
| iOS              | 13.1.1+    | Safari             | 13.7+           | ✓                       |
|                  |            | Chrome             | 86+             | ✓                       |
|                  |            | Firefox            | 33+             | ✓                       |
|                  |            | Microsoft Edge     | 89              | $\checkmark$            |
|                  |            | WeChat<br>embedded | -               | 1                       |
| Android          | -          | Chrome             | 86+             | ✓                       |
|                  |            | Firefox            | 88+             | ✓                       |
|                  |            | WeChat<br>embedded | X5 core         | 1                       |



|  | M/a Oh at |            |   |
|--|-----------|------------|---|
|  | wechat    | X Web core | ✓ |
|  | embedded  |            |   |
|  |           |            |   |

Additionally, in some browsers that support WebRTC, there may be decoding failures or server-side issues. In these cases, the player will convert the WebRTC URL to a more compatible HLS URL for playback. This behavior is known as downgrade processing.

To summarize, there are several scenarios that trigger downgrading:

The browser environment does not support WebRTC.

Failed to connect to the server, and the number of retries has exceeded the set value (internal status code -2004).

Decoding failure during playback (internal status code -2005).

Other WebRTC-related errors (internal status code -2001).

## Downgrade method

#### 1. Automatic downgrade

During player initialization, the Live Event Broadcasting address is passed through the sources field. In environments requiring downgrade processing, the player automatically converts the protocol, converting the Live Event Broadcasting address to an HLS protocol address.

For example, Live Event Broadcasting address:





webrtc://global-lebtest-play.myqcloud.com/live/lebtest?txSecret=f22a813b284137ed10d

It will automatically convert to:





https://global-lebtest-play.myqcloud.com/live/lebtest.m3u8?txSecret=f22a813b284137e

#### 2. Specified downgrade

In Adaptive Bit Rate (ABR) playback scenarios, if downgrading is necessary, one cannot simply convert formats to obtain the adaptive bitrate HLS address; it must be manually specified. Or in other scenarios where the user wishes to manually specify, downgrade addresses can be set in the following manner. The address is not limited to HLS protocol; it can also be of other protocols:





```
var player = TCPlayer('player-container-id',{
    sources: 'webrtc://global-lebtest-play.myqcloud.com/live/lebtest?txSecret=f22a8
    webrtcConfig: {
        fallbackUrl: 'https://global-lebtest-play.myqcloud.com/live/lebtest_HLSABR.m3
    },
});
```

## Downgrade callback

When a downgrade is triggered, the player will initiate a callback:



```
player.on('webrtcfallback', function(event) {
    console.log(event);
});
```

## Downgrade scenarios



Live Event Broadcasting is based on WebRTC and depends on the operating system and browser support for WebRTC.

Currently, the SDK has been tested on the following operating systems and browsers, with the test results as follows:

| Operating system | OS Version | Browser         | Browser version | Support for stream pull |
|------------------|------------|-----------------|-----------------|-------------------------|
| Windows          | win 10     | Chrome          | 86+             | 1                       |
|                  |            | Firefox         | 88+             | 1                       |
|                  |            | Microsoft Edge  | 86+             | 1                       |
| macOS 10         | 10.5+      | Safari          | 13.1+           | 1                       |
|                  |            | Chrome          | 86+             | 1                       |
|                  |            | Firefox         | 88+             | 1                       |
|                  |            | Microsoft Edge  | 86+             | 1                       |
| iOS 13.1.        | 13.1.1+    | Safari          | 13.7+           | 1                       |
|                  |            | Chrome          | 86+             | 1                       |
|                  |            | Firefox         | 33+             | 1                       |
|                  |            | Microsoft Edge  | 89              | 1                       |
|                  |            | WeChat embedded | -               | 1                       |
| Android          | -          | Chrome          | 86+             | $\checkmark$            |
|                  |            | Firefox         | 88+             | 1                       |
|                  |            | WeChat embedded | X5 core         | 1                       |
|                  |            | WeChat embedded | XWeb core       | 1                       |

Additionally, in some browsers that support WebRTC, there may be decoding failures or server-side issues. In these cases, the player will convert the WebRTC URL to a more compatible HLS URL for playback. This behavior is known as downgrade processing.

To summarize, there are several scenarios that trigger downgrading:

The browser environment does not support WebRTC.

Failed to connect to the server, and the number of retries has exceeded the set value (internal status code -2004).

Decoding failure during playback (internal status code -2005).

Other WebRTC-related errors (internal status code -2001).

## Downgrade method

#### 1. Automatic downgrade

During player initialization, the Live Event Broadcasting address is passed through the sources field. In environments requiring downgrade processing, the player automatically converts the protocol, converting the Live Event Broadcasting address to an HLS protocol address.

For example, Live Event Broadcasting address:



webrtc://global-lebtest-play.myqcloud.com/live/lebtest?txSecret=f22a813b284137ed10d



It will automatically convert to:



https://global-lebtest-play.myqcloud.com/live/lebtest.m3u8?txSecret=f22a813b284137e

#### 2. Specified downgrade

In Adaptive Bit Rate (ABR) playback scenarios, if downgrading is necessary, one cannot simply convert formats to obtain the adaptive bitrate HLS address; it must be manually specified. Or in other scenarios where the user wishes to manually specify, downgrade addresses can be set in the following manner. The address is not limited to HLS protocol; it can also be of other protocols:





```
var player = TCPlayer('player-container-id',{
    sources: 'webrtc://global-lebtest-play.myqcloud.com/live/lebtest?txSecret=f22a8
    webrtcConfig: {
        fallbackUrl: 'https://global-lebtest-play.myqcloud.com/live/lebtest_HLSABR.m3
    },
});
```

## Downgrade callback

When a downgrade is triggered, the player will initiate a callback:



```
player.on('webrtcfallback', function(event) {
    console.log(event);
```

});

## iOS Integration Integration Guide

Last updated : 2024-04-26 11:09:31

This document describes how to quickly integrate RT-Cube's LiteAVSDK\_Player for iOS into your project.

## **Environment Requirements**

Xcode 9.0 or later iPhone or iPad with iOS 9.0 or later A valid developer signature for your project

## Integrating the SDK

You can use CocoaPods to automatically load the SDK or manually download the SDK and import it into your project.

#### Integration via CocoaPods

#### 1. Install CocoaPods.

Enter the following command in a terminal window (you need to install Ruby on your Mac first):





sudo gem install cocoapods

#### 2. Create a Podfile.

Go to the directory of your project and enter the following command to create a Podfile in the directory.



pod init

#### 3. Edit the Podfile.

Use CocoaPod's official source, which allows version selection. Edit the Podfile: Directly integrate the latest version of TXLiteAVSDK\_Player\_Premium as a Pod:



```
platform :ios, '9.0'
source 'https://github.com/CocoaPods/Specs.git'
target 'App' do
pod 'TXLiteAVSDK_Player_Premium'
end
```

To specify a version, you can add the following dependency to the podfile file:





pod 'TXLiteAVSDK\_Player\_Premium', '~> 110.8.29000'

#### 4. Update the local repository and install the SDK.

Enter the following command in a terminal window to update the local repository file and install LiteAVSDK:



pod install

Or, run this command to update the local repository:





pod update

An XCWORKSPACE project file integrated with LiteAVSDK will be generated. Double-click to open the file.

## Manual SDK integration

1. Download the package of the SDK and demo on the latest version of TXLiteAVSDK\_Player\_Premium.

2. Add SDK/TXLiteAVSDK\_Player\_Premium.framework to the project to be integrated and select **Do Not Embed**.



3. You need to configure -ObjC of the project target; otherwise, the SDK will crash as the SDK category cannot be loaded.



Open Xcode, select the target, select the \*\*Build Settings\*\* tab, search for "Other

4. Add library files (in the SDK directory)

TXFFmpeg.xcframework: Add the .xcframework file to the project, set it to Embed & Sign in General > Frameworks, Libraries, and Embedded Content, and check whether Code Sign On Copy is selected in Build Phases > Embed Frameworks in your project settings as shown below:



TXSoundTouch.xcframework: Add the .xcframework file to the project, set it to Embed & Sign in General > Frameworks, Libraries, and Embedded Content, and check whether Code Sign On Copy is selected in Build Phases > Embed Frameworks in your project settings as shown below:

| <ul> <li>Embed Frameworks (2 items)</li> </ul> |                         |  |
|--|-------------------------|--|
| Destination                                    | Frameworks              |  |
| Subpath  |                         |  |
| Copy only when installing                      |                         |  |
| Name   |                         |  |
| 🚘 TXFFmpe                                      | g.xcframeworkin/SDK     |  |
| 🚘 TXSound                                      | Fouch.xcframeworkin/SDK |  |
|  |                         |  |
| + -  |                         |  |

Then, select **Build Settings** > **Search Paths** in Xcode and add the path of the above frameworks in **Framework S** earch Paths.

MetalKit.framework: Open Xcode, go to your project settings, select Build Phases > Link Binary With Libraries, click + in the bottom-left corner, and enter "MetalKit" to add it to the project as shown below:

| Choose frameworks and libraries to add:  |  |  |
|--|--|--|
| Q metal 8                                |  |  |
| 🛃 TXLiteAVDemo Project                   |  |  |
| Workspace                                |  |  |
| ✓ 📚 iOS 15.2                             |  |  |
| IibswiftMetal.tbd                        |  |  |
| IibswiftMetalKit.tbd                     |  |  |
| 🚘 Metal.framework                        |  |  |
| 🚔 MetalKit.framework                     |  |  |
| 🚘 MetalPerformanceShaders.framework      |  |  |
| 🚘 MetalPerformanceShadersGraph.framework |  |  |
| Developer Frameworks                     |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Add Other  Cancel Add                    |  |  |

**ReplayKit.framework**: Open Xcode, go to your **project settings**, select **Build Phases** > **Link Binary With Libraries**, click + in the bottom-left corner, and enter "ReplayKit" to add it to the project as shown below:

| <ul> <li>Link Binary With Libraries</li> </ul> |                           |
|--|---------------------------|
|  | Name                      |
|  | 🚔 MetalKit.framework      |
|  | 🚔 ReplayKit.framework     |
| a fallowing avetage                            | librarias in the same way |

Add the following system libraries in the same way:

**System frameworks:** SystemConfiguration, CoreTelephony, VideoToolbox, CoreGraphics, AVFoundation, Accelerat e, and MobileCoreServices.

System libraries: libz, libresolv, libiconv, libc++, and libsqlite3.



#### Picture-in-picture (PiP) feature

To use the PiP feature, configure as shown below. If you don't need the PiP feature, skip this part.

- 1. To use the PiP feature of iOS, upgrade the SDK to 10.3 or later.
- 2. To use the PiP feature, you need to enable the background mode. In Xcode, select the target, click Signing &

Capabilities > Background Modes, and select Audio, AirPlay, and Picture in Picture as shown below:

| P Background Modes     Background Modes     Section     Se |   |
|--|---|
|  | Modes 🗸 Audio, AirPlay, and Picture in Picture<br>Location updates<br>Voice over IP |

## Importing the SDK

There are two ways to import the SDK in your project code.

Method 1: import the SDK module in the files that need to use the SDK's APIs in your project





@import TXLiteAVSDK\_Player\_Premium;
// If you are using the basic version of the player, please use: @import TXLiteAVSD

Method 2: import a specific header file in the files that need to use the SDK's APIs in your project





#import "TXLiteAVSDK\_Player\_Premium/TXLiteAVSDK.h"
// If you are using the basic version of the player, please use: #import "TXLiteAVS

## **Configuring License**

1. Enter VOD console. to apply for a trial license as instructed in Adding and Renewing a License. If no license is configured, video playback will fail. You will get two strings: a license URL and a decryption key.

2. After obtaining the License information, you need to initialize and configure the License before calling the relevant interfaces of the SDK. For detailed tutorials, please see Configuring View License.

## FAQs

# What should I do if a duplicate symbol error occurs because my project integrates multiple editions of LiteAVSDK such as CSS, TRTC, and Player?

If you integrate two or more editions of LiteAVSDK (MLVB, Player, TRTC, UGSV), a library conflict error will occur when you build your project. This is because some symbol files are shared among the underlying libraries of the SDKs. To solve the problem, we recommend you integrate the All-in-One SDK, which includes the features of MLVB, Player, TRTC, and UGSV. For details, see SDK Download.

## How to call the API method of the SDK in Swift project?

If you want to call the API interface of the SDK in the Swift project, there are two ways:

### Method 1: Using a bridging header file

1. Create a bridging header file, for example, \*\*\*-Bridging-Header.h, and add the following code: #import

<TXLiteAVSDK\_Player\_Premium/TXLiteAVSDK.h>.

2. Configure the Objective-c Bridging header option in the Build Setting of the project. Set the path of the bridging file and add it to Objective-c Bridging header (e.g., \$(SRCROOT)/SwiftCallOC/\*\*\*-Bridging-Header.h, depending on the specific path of the project). Compile and run the project.

## Method 2: Using the module.modulemap file in the SDK

1. Check if the TXLiteAVSDK\_Player\_Premium.framework contains the Modules - module.modulemap file (Player SDK provides it by default).

2. Configure the Swift Compiler - Search Paths option in the Build Setting of the project. Add the directory path where the module.modulemap file is located or the parent directory path. Here, it can be:

\${PODS\_ROOT}/TXLiteAVSDK\_Player\_Premium/TXLiteAVSDK\_Player\_Premium/TXLiteAVSDK\_Player\_Premium.f ramework/Modules (depending on the specific path of the project).

3. At the top of the class where you need to call the method, use import TXLiteAVSDK\_Player\_Premium to import and call the relevant methods.

For the above integration methods and demo, please refer to the GitHub demo for details.

## **VOD** Scenario

Last updated : 2024-08-07 15:15:10

## Limits

- 1. Activate VOD. If you don't have an account yet, sign up for one first.
- 2. Download and install Xcode from App Store.
- 3. Download and install CocoaPods as instructed at the CocoaPods website.

## This Document Describes

How to integrate the Tencent Cloud Player SDK for iOS. How to use the Player SDK for VOD playback. How to use the underlying capabilities of the Player SDK to implement more features.

## **SDK** Integration

### Step 1. Integrate the SDK ZIP file

Download and integrate the SDK ZIP file as instructed in Integration Guide.

### Step 2. Configure the license

If you have obtained a license, you can view the license URL and key in the VOD console.

If you don't have the required license yet, you can get it as instructed in Adding and Renewing a License.

After obtaining the License information, you need to initialize and configure the License before calling the relevant interfaces of the SDK. For detailed tutorials, please see Configuring View License.

### Step 3. Create a player object

The TXVodPlayer module of the Player SDK is used to implement the VOD feature.





TXVodPlayer \*\_txVodPlayer = [[TXVodPlayer alloc] init]; [\_txVodPlayer setupVideoWidget:\_myView insertIndex:0]

#### Step 4. Create a rendering view

In iOS, a view is used as the basic UI rendering unit. Therefore, you need to configure a view, whose size and position you can adjust, for the player to display video images on.





#### [\_txVodPlayer setupVideoWidget:\_myView insertIndex:0]

Technically, the player does not render video images directly on the view ( \_myView in the sample code) you provide. Instead, it creates a subview for OpenGL rendering over the view.

You can adjust the size of video images by changing the size and position of the view. The SDK will make changes to the video images accordingly.





#### How to make an animation

You are allowed great flexibility in view animation, but note that you need to modify the transform rather than

frame attribute of the view.





```
[UIView animateWithDuration:0.5 animations:^{
    _myView.transform = CGAffineTransformMakeScale(0.3, 0.3); // Shrink by 1/3
}];
```

## Step 5. Start playback

TXVodPlayer supports two playback modes for you to choose as needed:

Through URL

Through `fileId`



TXVodPlayer will internally recognize the playback protocol automatically. You only need to pass in your playback URL to the startPlay function.



```
// Play back a video resource at a URL
NSString* url = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
[_txVodPlayer startVodPlay:url];
// Play back a local video resource in the sandbox
// Get the `Documents` path
NSString *documentPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
// Get the local video path
```



NSString \*videoPath = [NSString stringWithFormat:@"%@/video1.m3u8",documentPath];
[\_txVodPlayer startVodPlay:videoPath];



```
TXPlayerAuthParams *p = [TXPlayerAuthParams new];
p.appId = 1252463788;
p.fileId = @"4564972819220421305";
// `psign` is a player signature. For more information on the signature and how to
p.sign = @"psignxxxxx"; // The player signature
[_txVodPlayer startVodPlayWithParams:p];
```



You can go to Media Assets and find it. After clicking it, you can view its fileId in the video details on the right. Play back the video through the fileId, and the player will request the backend for the real playback URL. If the network is abnormal or the fileId doesn't exist, the PLAY\_ERR\_GET\_PLAYINFO\_FAIL event will be received; otherwise, PLAY\_EVT\_GET\_PLAYINFO\_SUCC will be received, indicating that the request succeeded.

#### Step 6. Stop playback

Remember to use **removeVideoWidget** to terminate the view control before exiting the current UI when stopping playback. This can prevent memory leak and screen flashing issues.



// Stop playback



```
[_txVodPlayer stopPlay];
[_txVodPlayer removeVideoWidget]; // Remember to terminate the view control
```

## **Basic Feature Usage**

1. Playback control

Starting playback




```
// Start playback
[_txVodPlayer startVodPlay:url];
```

#### Pausing playback



// Pause the video
[\_txVodPlayer pause];

**Resuming playback** 



// Resume the video
[\_txVodPlayer resume];

# Stopping playback



```
// Stop the video
[_txVodPlayer stopPlay];
```

## Adjusting playback progress (seek)

When the user drags the progress bar, seek can be called to start playback at the specified position. The Player SDK supports accurate seek.





int time = 600; // In seconds if the value is of `int` type
// Adjust the playback progress
[\_txVodPlayer seek:time];

#### Precise and Imprecise Seek

Starting from version 11.8 of the player SDK, it is supported to specify precise or imprecise seek when calling the seek interface.





```
float time = 600; // float type, unit is seconds
// Adjust progress
[_txVodPlayer seek:time accurateSeek:YES]; // Accurate seek
[_txVodPlayer seek:time accurateSeek:NO]; // Non-accurate seek
```

#### Seek to the specified Program Date Time (PDT) point in the video stream

To seek to the specified Program Date Time (PDT) point in the video stream, which enables functions such as fastforward, rewind, and progress bar jumping, currently only HLS video format is supported. Note: Starting from version 11.6 of the player's advanced edition, this function is supported.

©2013-2022 Tencent Cloud. All rights reserved.



long long pdtTimeMs = 600; // Unit is milliseconds
[\_txVodPlayer seekToPdtTime:time];

## Specifying playback start time

You can specify the playback start time before calling startVodPlay for the first time.





```
float startTimeInSecond = 60; // Unit: Second
[_txVodPlayer setStartTime:startTimeInSecond]; // Set the playback start time
[_txVodPlayer startVodPlay:url];
```

#### 2. Image adjustment

#### view: size and position

You can modify the size and position of the view by adjusting the size and position of the parameter view of setupVideoWidget. The SDK will automatically adjust the size and position of the view based on your configuration.

#### setRenderMode: Aspect fill or aspect fit

| Value                   | Description  |
|-------------------------|--|
| RENDER_MODE_FILL_SCREEN | Images are scaled to fill the entire screen, and the excess parts are cropped. There are no black bars in this mode, but images may not be displayed in whole. |
| RENDER_MODE_FILL_EDGE   | Images are scaled as large as the longer side can go. Neither side exceeds the screen after scaling. Images are centered, and there may be black bars.         |

#### setRenderRotation: Image rotation

| Value                  | Description  |
|------------------------|--|
| HOME_ORIENTATION_RIGHT | The Home button is on the right of the video image |
| HOME_ORIENTATION_DOWN  | The Home button is below the video image           |
| HOME_ORIENTATION_LEFT  | The Home button is on the left of the video image  |
| HOME_ORIENTATION_UP    | The Home button is above the video image           |

# 3. Adjustable-Speed playback

The VOD player supports adjustable-speed playback. You can use the setRate API to set the VOD playback speed, such as 0.5x, 1.0x, 1.2x, and 2x speed.



```
// Set playback at 1.2X rate
[_txVodPlayer setRate:1.2];
// Start playback
[_txVodPlayer startVodPlay:url];
```

# 4. Playback loop



// Set playback loop
[\_txVodPlayer setLoop:true];
// Get the current playback loop status
[\_txVodPlayer loop];

# 5. Muting/Unmuting





// Mute or unmute the player. true: Mute; false: Unmute
[\_txVodPlayer setMute:true];

# 6. Screencapturing

Call **snapshot** to take a screenshot of the current video frame. This method captures only the video frame. To capture the UI, use the corresponding API of the iOS system.

# 7. Roll image ad

The Player SDK allows you to add roll images on the UI for advertising as follows:

If autoPlay is set to NO, the player will load the video normally but will not immediately start playing it back. Users can see the roll image ad on the player UI after the player is loaded and before the video playback starts. When the ad display stop conditions are met, the resume API will be called to start video playback.

## 8. HTTP-REF

headers in TXVodPlayConfig can be used to set HTTP request headers, such as the Referer field commonly used to prevent the URL from being copied arbitrarily (Tencent Cloud provides a more secure signature-based hotlink protection solution) and the Cookie field for client authentication.





```
NSMutableDictionary<NSString *, NSString *> *httpHeader = [[NSMutableDictionary all
[httpHeader setObject:@"${Referer Content}" forKey:@"Referer"];
[_config setHeaders:httpHeader];
[_txVodPlayer setConfig:_config];
```

## 9. Hardware acceleration

It is extremely difficult to play back videos of the Blu-ray (1080p) or higher image quality smoothly if only software decoding is used. Therefore, if your main scenario is game live streaming, we recommend you use hardware acceleration.

Before switching between software and hardware decoding, you need to call **stopPlay** first. After the switch, you need to call **startVodPlay**; otherwise, severe blurs will occur.



```
[_txVodPlayer stopPlay];
_txVodPlayer.enableHWAcceleration = YES;
[_txVodPlayer startVodPlay:_flvUrl type:_type];
```

# **10. Definition settings**

The SDK supports the multi-bitrate format of HLS, so users can switch between streams at different bitrates. You can get the array of multiple bitrates as follows:





```
NSArray *bitrates = [_txVodPlayer supportedBitrates]; // Get the array of multiple
// TXBitrateItem class field meaning: index-bitrate subscript; width-video width; h
TXBitrateItem *item = [bitrates objectAtIndex:i];
[_txVodPlayer setBitrateIndex:item.index]; // Switch bit rate to desired definition
```

// Get the bit rate subscript of the current playback, the return value -1000 is th
int index = [\_txVodPlayer bitrateIndex];

During playback, you can call - [TXVodPlayer setBitrateIndex:] at any time to switch the bitrate. During switch, the data of another stream will be pulled. The SDK is optimized for Tencent Cloud multi-bitrate files to



implement smooth switch.

If you know the resolution information of the video stream in advance, you can specify the resolution of the video to be played before starting the broadcast, so as to avoid switching the stream after playback. For detailed methods, refer to Player configuration#Specify resolution before starting broadcast.

# 11. Adaptive bitrate streaming

The SDK supports adaptive bitrate streaming of HLS. After this capability is enabled, the player can dynamically select the most appropriate bitrate for playback based on the current bandwidth. You can enable adaptive bitrate streaming as follows:



```
[_txVodPlayer setBitrateIndex:-1]; // Pass in `-1` for the `index` parameter
```

During playback, you can call - [TXVodPlayer setBitrateIndex:] at any time to switch to another bitrate. After the switch, adaptive bitrate streaming will be disabled.

# 12. Enabling smooth bitrate switch

Before starting playback, you can enable smooth bitrate switch to seamlessly switch between different definitions (bitrates) during playback. If smooth bitrate switch is enabled, the transition between different bitrates will be smoother but will be more time-consuming. Therefore, this feature can be configured as needed.



```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
// If it is set to `YES`, the bitrate can be switched smoothly when IDR frames are
[_config setSmoothSwitchBitrate:YES];
[_txVodPlayer setConfig:_config];
```

## 13. Playback progress listening

There are two metrics for the VOD progress: **loading progress** and **playback progress**. Currently, the SDK notifies the two progress metrics in real time through event notifications. For more information on the event notification content, see Event Listening.





```
-(void) onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary*
    if (EvtID == PLAY_EVT_PLAY_PROGRESS) {
        // Loading progress in seconds. The decimal part is in ms.
        float playable = [param[EVT_PLAYABLE_DURATION] floatValue];
        [_loadProgressBar setValue:playable];
        // Playback progress in seconds. The decimal part is in ms.
        float progress = [param[EVT_PLAY_PROGRESS] floatValue];
        [_seekProgressBar setValue:progress];
        // Total video duration in seconds. The decimal part is in ms.
        float duration = [param[EVT_PLAY_DURATION] floatValue];
        // Total video duration in seconds. The decimal part is in ms.
        float duration = [param[EVT_PLAY_DURATION] floatValue];
        // It can be used to set duration display, etc.
        // Get the PDT time. This function is supported starting from version 1
        long long pdt_time_ms = [param[VOD_PLAY_EVENT_PLAY_PDT_TIME_MS] longLon
    }
}
```

#### 14. Playback network speed listening

You can display the current network speed when the video is lagging by listening on events.

You can use the NET\_SPEED of onNetStatus to get the current network speed. For detailed directions, see Playback status feedback (onNetStatus).

After the **PLAY\_EVT\_PLAY\_LOADING** event is detected, the current network speed will be displayed.

After the PLAY\_EVT\_VOD\_LOADING\_END event is received, the view showing the current network speed will be hidden.

#### 15. Video resolution acquisition

The Player SDK plays back a video through a URL string. The URL doesn't contain the video information, and you need to access the cloud server to load such information. Therefore, the SDK can only send the video information to your application as event notifications. For more information, see Event Listening.

#### **Resolution information**

Method 1

Method 2

Use the VIDEO\_WIDTH and VIDEO\_HEIGHT of onNetStatus to get the video width and height. For detailed directions, see Status feedback (onNetStatus).

Directly call -[TXVodPlayer width] and -[TXVodPlayer height] to get the current video width and height.

#### 16. Player buffer size

During normal video playback, you can control the maximum size of the data buffered from the network in advance. If the maximum buffer size is not configured, the player will use the default buffer policy to guarantee a smooth playback experience.



```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setMaxBufferSize:10]; // Maximum buffer size during playback in MB
[_txVodPlayer setConfig:_config]; // Pass in `config` to `_txVodPlayer`
```

## 17. Local video cache

In short video playback scenarios, the local video file cache is required, so that general users don't need to consume traffic again to reload an already watched video.

Supported format: The SDK supports caching videos in two common VOD formats: HLS (M3U8) and MP4.

**Enablement time:** The SDK doesn't enable the caching feature by default. We recommend you do not enable it for scenarios in which most videos are watched only once.

**Enablement method:** To enable it, you need to configure two parameters: local cache directory and cache size.



// Set the global cache directory of the playback engine
NSArray \*paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDom
NSString \*documentsDirectory = [paths objectAtIndex:0];

#### Note:

The TXVodPlayConfig#setMaxCacheItems API used for configuration on earlier versions has been deprecated and is not recommended.

## 18.Screen control (screen on and off)

Due to the frequent personalized settings in mobile phones, the screen lock time is often set, which may cause the screen to turn off (or be locked) during video playback, greatly affecting the user experience. Therefore, to solve this problem, the following code needs to be added at relevant times during the playback process to keep the screen always on.





```
(1) Bright screen (no screen off)
// start playing (startVodPlay / startPlayDrm / startVodPlayWithParams)
// resume playback (resume)
[[UIApplication sharedApplication] setIdleTimerDisabled:YES];
(2) Off screen (restore off screen)
// stop (stopPlay)
// pause (pause)
[[UIApplication sharedApplication] setIdleTimerDisabled:NO];
```

#### Note:

Please pay attention to calling the above interfaces in the main thread.

# **19.DRM encrypted video playback**

#### Note:

This feature requires the premium version of the player to be supported.

The advanced version of the player SDK supports playback of commercial-grade DRM-encrypted videos, and currently supports two DRM schemes, WideVine and Fairplay. For more commercial-grade DRM information, please refer to the product introduction.

DRM-encrypted videos can be played in the following 2 ways:

Play by FiledId

Custom configuration play





```
TXPlayerAuthParams *p = [TXPlayerAuthParams new];
p.appId = ${appId}; // appId of the Tencent Cloud account
p.fileId = @"${fieId}"; // fileId of DRM encrypted video
// psign is the signature of the player. Please refer to the link for signature int
p.sign = @"${psgin}"; // Player signature for encrypted video
[_txVodPlayer startVodPlayWithParams:p];
```

Playing via FileId is suitable for accessing the cloud on-demand background. This method is no different from playing ordinary FileId files. You need to configure the resource as a DRM type in Cloud VOD, and the SDK will recognize and process it internally.





// Play through the TXVodPlayer#startPlayDrm interface // @param certificateUrl certificate provider url // @param licenseUrl decrypted key url // @param videoUrl Url address of the video to be played TXPlayerDrmBuilder \*builder = [[TXPlayerDrmBuilder alloc] initWithDeviceCertificate [\_txVodPlayer startPlayDrm:builder];

#### **20.External subtitles**

Note:

This feature requires the premium version of the player to be supported.

The advanced version of the player SDK supports adding and switching external subtitles, and now supports subtitles in two formats: SRT and VTT.

Best practice: It is recommended to add subtitles and configure subtitle styles before calling startVodPlay. After receiving the PLAY\_EVT\_VOD\_PLAY\_PREPARED event, call selectTrack to choose the subtitle. Adding subtitles does not automatically load them. After calling selectTrack, the subtitles will be loaded. The successful selection of subtitles will trigger the VOD\_PLAY\_EVT\_SELECT\_TRACK\_COMPLETE event callback. The usage is as follows :

#### Step 1: Add external subtitles





// Pass in subtitle url, subtitle name, subtitle type. It is recommended to add sub
[\_txVodPlayer addSubtitleSource:@"https://mediacloud-76607.gzc.vod.tencent-cloud.co

Step 2: Switch subtitles after playback.





```
// After starting to play the video, select the added external subtitles. Please ca
NSArray<TXTrackInfo *> *subtitlesArray = [_txVodPlayer getSubtitleTrackInfo];
for (int i = 0; i < subtitlesArray.count; i++) {
    TXTrackInfo *info = subtitlesArray[i];
    if (info.trackIndex == 0) {
        [_txVodPlayer selectTrack:info.trackIndex]; // check subtitles
    } else {
        // If other subtitles are not needed, perform deselectTrack
        [_txVodPlayer deselectTrack:info.trackIndex];
    }
}
```

```
// Listen for track switch messages
- (void)onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary
    if (EvtID == VOD_PLAY_EVT_SELECT_TRACK_COMPLETE) {
        int trackIndex = [(NSNumber *)[param valueForKey:EVT_KEY_SELECT_TRACK_INDEX
        int errorCode = [(NSNumber *)[param valueForKey:EVT_KEY_SELECT_TRACK_ERROR_
        NSLog(@"receive VOD_PLAY_EVT_SELECT_TRACK_COMPLETE, trackIndex=%d, errorCo
    }
}
```

# Step 3: Configure subtitle style.

The subtitle style supports configuration before or during playback.





// For detailed parameter configuration, please refer to the API documentation
TXPlayerSubtitleRenderModel \*model = [[TXPlayerSubtitleRenderModel alloc] init];
model.canvasWidth = 1920; // The width of the subtitle rendering canvas
model.canvasHeight = 1080; // The height of the subtitle rendering canvas
model.isBondFontStyle = NO; // Set whether the subtitle font is bold
model.fontColor = 0xFF000000; // Set the subtitle font color, the default is white
[\_txVodPlayer setSubtitleStyle:model];

## 21.Switching between multiple audio tracks



The advanced version of the player SDK supports switching between multiple audio tracks built into the video. The usage is as follows:



```
NSArray<TXTrackInfo *> *soundTrackArray = [_txVodPlayer getAudioTrackInfo];
for (int i = 0; i < soundTrackArray.count; i++) {
    TXTrackInfo *info = soundTrackArray[i];
    if (info.trackIndex == 0) {
        // Switch to the desired audio track by determining the trackIndex or name.
        [_txVodPlayer selectTrack:info.trackIndex];
    } else {
        // If other subtitles are not required, proceed with deselectTrack.
```

```
[_txVodPlayer deselectTrack:info.trackIndex];
}
```

# Using Advanced Features

# 1. Video preloading

## Step 1. Use video preloading

In UGSV playback scenarios, the preloading feature contributes to a smoother viewing experience: While watching a video, you can load the URL of the next video to be played back on the backend. When the next video is switched to, it will be preloaded and can be played back immediately.

Video preloading can deliver an instant playback effect but has certain performance overheads. If your business needs to preload many videos, we recommend you use this feature together with video predownloading. This is how seamless switch works in video playback. You can use <code>isAutoPlay</code> in <code>TXVodPlayer</code> to implement the feature as follows:







```
// Play back video A: If `isAutoPlay` is set to `YES`, the video will be immediatel
NSString* url_A = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f10.mp4";
_player_A.isAutoPlay = YES;
[_player_A startVodPlay:url_A];
```

```
// To preload video B when playing back video A, set `isAutoPlay` to `NO`
NSString* url_B = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
_player_B.isAutoPlay = NO;
[_player_B startVodPlay:url_B];
```



After video A ends and video B is automatically or manually switched to, you can call the resume function to immediately play back video B.

#### Note:

After autoPlay is set to false, make sure that video B has been prepared before calling resume, that is, you should call it only after the PLAY\_EVT\_VOD\_PLAY\_PREPARED event of video B (2013: the player has been prepared, and the video can be played back) is detected.



```
if (EvtID == PLAY_EVT_PLAY_END) {
    [_player_A stopPlay];
    [_player_B setupVideoWidget:mVideoContainer insertIndex:0];
    [_player_B resume];
    }
}
```

# Step 2. Configure the video preloading buffer

You can set a large buffer to play back videos more smoothly under unstable network conditions. You can set a smaller buffer to reduce the traffic consumption.

## Preloading buffer size

This API is used to control the maximum buffer size before the playback starts in preloading scenarios (that is, AutoPlay of the player is set to false before video playback starts).




```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setMaxPreloadSize:(2)];; // Maximum preloading buffer size in MB. Set it
[_txVodPlayer setConfig:_config]; // Pass in `config` to `_txVodPlayer`
```

#### Playback buffer size

During normal video playback, you can control the maximum size of the data buffered from the network in advance. If the maximum buffer size is not configured, the player will use the default buffer policy to guarantee a smooth playback experience.





```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setMaxBufferSize:10]; // Maximum buffer size during playback in MB
[_txVodPlayer setConfig:_config]; // Pass in `config` to `_txVodPlayer`
```

# 2. Video predownloading

You can download part of the video content in advance without creating a player instance, so as to start playing back the video faster when using the player. This helps deliver a better playback experience. Before using the playback service, make sure that video cache has been set.



#### Note:

TXPlayerGlobalSetting is the global cache setting API, and the original TXVodConfig API has been deprecated.

The global cache directory and size settings have a higher priority than those configured in TXVodConfig of the player.

Pre-download by media URL

Pre-download by media filedId

An example code for pre-downloading a video via a media asset URL is as follows:



// Set the global cache directory of the playback engine



```
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDom
NSString *documentsDirectory = [paths objectAtIndex:0];
NSString *preloadDataPath = [documentsDirectory stringByAppendingPathComponent:@"/p
if (![[NSFileManager defaultManager] fileExistsAtPath:preloadDataPath]) {
    [[NSFileManager defaultManager] createDirectoryAtPath:preloadDataPath
                              withIntermediateDirectories:NO
                                               attributes:nil
                                                    error:&error]; //Create folder
}
[TXPlayerGlobalSetting setCacheFolderPath:preloadDataPath];
// Set the playback engine cache size
[TXPlayerGlobalSetting setMaxCacheSize:200];
NSString *m3u8url = "http://****";
int taskID = [[TXVodPreloadManager sharedManager] startPreload:m3u8url
                                                   preloadSize:10
                                           preferredResolution:1920*1080
                                                      delegate:self];
// Cancel predownloading
[[TXVodPreloadManager sharedManager] stopPreload:taskID];
```

#### Note:

Pre-download by fileId is supported since version 11.3.

Pre-downloading by fileId is a time-consuming operation, please do not call it in the main thread, otherwise an illegal call exception will be thrown. The preferredResolution passed in during startPreload must be consistent with the preferred resolution set when starting the broadcast, otherwise the expected effect will not be achieved. An example of use is as follows:





©2013-2022 Tencent Cloud. All rights reserved.



# 3. Video download

Video download allows users to download online videos and watch them offline. If the video is encrypted, the downloaded video through the player SDK will be kept in an encrypted state locally and can only be decrypted and played through Tencent Cloud Player SDK. This can effectively prevent illegal dissemination of downloaded videos and protect video security.

As HLS streaming media cannot be directly saved locally, you cannot download them and play back them as local files. You can use the video download scheme based on TXVodDownloadManager to implement offline HLS playback.

# Note:

Currently, TXVodDownloadManager can cache only HLS files but not MP4 and FLV files.

The Player SDK already supports playing back local MP4 and FLV files.

# Step 1. Make preparations

When the SDK is initialized, set the global storage path for functions such as video download, preload, and cache. The usage is as follows:





NSString \*cachesDir = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSU
NSString downloadPath = [NSString stringWithFormat:@"%@/txdownload",cachesDir];
[TXPlayerGlobalSetting setCacheFolderPath:downloadPath];

TXVodDownloadManager is designed as a singleton; therefore, you cannot create multiple download objects. It is used as follows:





TXVodDownloadManager \*downloader = [TXVodDownloadManager shareInstance];

#### Step 2. Start the download

You can start the download through the fileid or URL.

Through fileid

Through URL

You need to pass inappIdandfileIdat least for download throughfileid. If you don't specify a valueforuserName,defaultwill be used by default. Note: You can download encrypted videos only through



Fileid and must enter the psign parameter.



```
TXVodDownloadDataSource *source = [[TXVodDownloadDataSource alloc] init];
source.appId = 1252463788;
source.fileId = @"4564972819220421305";
// // `psign` is a player signature. For more information on the signature and how
source.pSign = @"xxxxxxxxxx";
// Specify the download definition
```

```
// Valid values: `TXVodQualityOD` (original), `TXVodQualityFLU` (LD), `TXVodQuality
source.quality = TXVodQualityHD; // HD
```



// \*\*Note that if you use the legacy v2 protocol for download, set the `appId` and
// source.auth = auth; \*\*There is no need to set it by default.\*\*

```
[downloader startDownload:dataSource];
```

You only need to pass in the download URL. Only the non-nested HLS, i.e., single-bitstream HLS, is supported. Use fileid in case of private encryption.



[downloader startDownloadUrl:@"http://1253131631.vod2.myqcloud.com/26f327f9vodgzp12



## Step 3. Receive the task information

Before receiving the task information, you need to set the callback delegate first.



downloader.delegate = self;

You may receive the following task callbacks:

| Callback Message                             | Description  |
|--|--|
| -[TXVodDownloadDelegate<br>onDownloadStart:] | The task started, that is, the SDK started the download. |
|  |  |

| -[TXVodDownloadDelegate<br>onDownloadProgress:]       | Task progress. During download, the SDK will frequently call back this API. You can update the displayed progress here.  |
|---|--|
| -[TXVodDownloadDelegate<br>onDownloadStop:]           | The task stopped. When you call stopDownload to stop the download, if this message is received, the download is stopped successfully.  |
| -[TXVodDownloadDelegate<br>onDownloadFinish:]         | Download was completed. If this callback is received, the entire file has been downloaded, and the downloaded file can be played back by `TXVodPlayer`.                            |
| -[TXVodDownloadDelegate<br>onDownloadError:errorMsg:] | A download error occurred. If the network is disconnected during download, this API will be called back and the download task will stop. For all error codes, see TXDownloadError. |

# Download error code

| error code            | value | Meaning  |
|-----------------------|-------|--|
| TXDownloadSuccess     | 0     | Download successful  |
| TXDownloadAuthFaild   | -5001 | Failed to request video information from the cloud on-demand console,<br>it is recommended to check whether the fileId and psign parameters are<br>correct |
| TXDownloadNoFile      | -5003 | No file for this resolution  |
| TXDownloadFormatError | -5004 | The download file format is not supported  |
| TXDownloadDisconnet   | -5005 | The network is disconnected, it is recommended to check whether the network is normal  |
| TXDownloadHIsKeyError | -5006 | Failed to get HLS decryption key   |
| TXDownloadPathError   | -5007 | Download directory access failed, it is recommended to check whether you have permission to access the download directory                                  |

As the downloader can download multiple files at a time, the callback API carries the

 TXVodDownloadMediaInfo
 object. You can access the URL or
 dataSource
 to determine the download

 source and get other information such as download progress and file size.

# Step 4. Stop the download

You can call the -[TXVodDownloadManager stopDownload:] method to stop the download. The parameter is the object returned by -[TXVodDownloadManager sartDownloadUrl:]. The SDK supports checkpoint

**restart.** If the download directory is not changed, when you resume downloading a file, the download will start from the point where it stopped.

# Step 5. Manage downloads

1. You can get the download lists of all accounts or the specified account.



```
// s a time-consuming function. Please do not call it in the main thread
NSArray<TXVodDownloadMediaInfo *> *array = [[[TXVodDownloadManager shareInstance] g
// Get the download list of the `default` user
for (TXVodDownloadMediaInfo *info in array) {
    if ([info.userName isEqualToString:@"default"]) {
```

# Stencent Cloud

```
// Save the download list of the `default` user
}
```

2. Get the download information of a FileId or URL:

2.1 To get the download information of a Fileid through the -[TXVodDownloadManager getDownloadMediaInfo:] API, such as the current download status and progress, you need to pass in AppID, Fileid, and qualityId.



```
// Get the download information of a `fileId`
TXVodDownloadMediaInfo *sourceMediaInfo = [[TXVodDownloadMediaInfo alloc] init];
```

```
TXVodDownloadDataSource *dataSource = [[TXVodDownloadDataSource alloc] init];
dataSource.appId = 1252463788;
dataSource.fileId = @"4564972819220421305";
dataSource.pSign = @"psignxxxx";
dataSource.quality = TXVodQualityHD;
sourceMediaInfo.dataSource = dataSource;
TXVodDownloadMediaInfo *downlaodMediaInfo = [[TXVodDownloadManager shareInstance]
// Get the total size of the file being downloaded in bytes. This API takes effect
// Note: The total size refers to the size of the original file uploaded to the VOD
downlaodMediaInfo.size; // Get the total size of the file being downloaded
downlaodMediaInfo.duration; // Get the total duration
downlaodMediaInfo.playableDuration; // Get the playable duration of the downloaded
downlaodMediaInfo.progress; // Get the download progress
downlaodMediaInfo.playPath; // Get the offline playback path, which can be passed
downlaodMediaInfo.downloadState; // Get the download status. For more information,
[downlaodMediaInfo isDownloadFinished]; // If `YES` is returned, the download is c
```

2.2 To get the download information of a URL, you simply need to pass in the URL information.





```
// Get the download information of a `fileId`
TXVodDownloadMediaInfo *sourceMediaInfo = [[TXVodDownloadMediaInfo alloc] init];
mediaInfo.url = @"videoURL";
TXVodDownloadMediaInfo *downlaodMediaInfo = [[TXVodDownloadManager shareInstance]
```

3. Delete the download information and relevant file:

If you don't need to resume the download, call the -[TXVodDownloadManager deleteDownloadFile:] method to delete the file to release the storage space.

#### Step 6: Play offline after downloading



The downloaded video can be played without internet connection, no internet connection is required. Once the download is complete, it can be played.



NSArray<TXVodDownloadMediaInfo \*> \*mediaInfoList = [[TXVodDownloadManager shareInst
TXVodDownloadMediaInfo \*mediaInfo = [mediaInfoList firstObject]; // Find the curren
if (mediaInfo.downloadState == TXVodDownloadMediaInfoStateFinish) { // Determine wh
 [self.player startVodPlay:mediaInfo.playPath];
}

Note:

When downloading and playing offline, be sure to get the download list and play it through the PlayPath of the download list video object TXVodDownloadMediaInfo, do not save the PlayPath object directly.

# 4. Encrypted playback

The video encryption solution is used in scenarios where the video copyright needs to be protected, such as online education. To encrypt your video resources, you need to alter the player and encrypt and transcode video sources. For more information, see Media Encryption and Copyright Protection Overview.

After you get the appId as well as the encrypted video's fileId and psign in the Tencent Cloud console, you can play back the video as follows:



```
TXPlayerAuthParams *p = [TXPlayerAuthParams new];
p.appId = 1252463788; // The `appId` of the Tencent Cloud account
p.fileId = @"4564972819220421305"; // The video's `fileId`
// `psign` is a player signature. For more information on the signature and how to
p.sign = @"psignxxxxx"; // The player signature
[_txVodPlayer startVodPlayWithParams:p];
```

# 5. Player configuration

Before calling statPlay, you can call setConfig to configure the player parameters, such as player connection timeout period, progress callback interval, and maximum number of cached files. TXVodPlayConfig allows you to configure detailed parameters. For more information, see TXVodPlayConfig. Below is the configuration sample code:





```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setEnableAccurateSeek:true]; // Set whether to enable accurate seek. Defa
[_config setMaxCacheItems:5]; // Set the maximum number of cached files to 5
[_config setProgressInterval:200]; // Set the progress callback interval in ms
[_config setMaxBufferSize:50]; // Set the maximum preloading buffer size in MB
[_txVodPlayer setConfig:_config]; // Pass in `config` to `_txVodPlayer`
```

Specifying resolution when playback starts

When playing back an HLS multi-bitrate video source, if you know the video stream resolution information in advance, you can specify the preferred resolution before playback starts, and the player will select and play back the stream at or below the preferred resolution. In this way, after playback starts, you don't need to call setBitrateIndex to switch to the required bitstream.



```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
// The parameter passed in is the product of the video width and height. You can pa
[_config setPreferredResolution:720*1280];
[_txVodPlayer setConfig:_config]; // Pass in `config` to `_txVodPlayer`
```



#### Specifying media type before playback

When the media type to be played is known in advance, the playback type detection within the player SDK can be reduced and the startup speed can be improved by configuring TXVodPlayConfig#setMediaType.

# Note:

TXVodPlayConfig#setMediaType is supported since version 11.2.



```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setMediaType:MEDIA_TYPE_FILE_VOD]; // Used to increase the speed of MP4 p
// [_config setMediaType:MEDIA_TYPE_HLS_VOD]; // Used to increase the speed of HLS
[_txVodPlayer setConfig:_config];
```



#### Setting playback progress callback interval



```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setProgressInterval:200]; // Set the progress callback interval in ms
[_txVodPlayer setConfig:_config]; // Pass in `config` to `_txVodPlayer`
```

# 6、HttpDNS resolution service

HTTPDNS is a domain name resolution service that sends domain name resolution requests to DNS servers based on the HTTP protocol, replacing the traditional method of sending resolution requests to the operator's local DNS



based on the DNS protocol. This method can avoid domain name hijacking and cross-network access issues caused by local DNS, and solve the problem of video playback failure caused by abnormal domain name resolution in mobile internet services.

## Note:

HttpDNS resolution service is supported from version 10.9 onwards.

1. To enable HTTPDNS resolution service

you can choose Tencent Cloud or other cloud providers and open the service. Make sure to integrate the service into the playback SDK after successful activation.

2. Accessing HTTPDNS Resolution Service in the Playback SDK

Taking Tencent Cloud HTTPDNS as an example, the following steps demonstrate how to access the service in the player SDK:





```
// Step 1: Turn on the HttpDNS resolution switch.
[TXLiveBase enableCustomHttpDNS:YES];
// Step 2: Implement HttpDNS resolution proxy: TXLiveBaseDelegate#onCustomHttpDNS.
- (void)onCustomHttpDNS:(NSString *)hostName ipList:(NSMutableArray<NSString *> *)1
// After resolving the hostName to an IP address, save it to the iPList and return
// MSDKDnsResolver is the HTTPDNS SDK resolution interface provided by Tencent Clou
NSArray *result = [[MSDKDns sharedInstance] WGGetHostByName:hostName];
NSString *ip = nil;
if (result && result.count > 1) {
if (![result[1] isEqualToString:@"0"]) {
ip = result[1];
```

```
} else {
ip = result[0];
}
[list addObject:ip];
}
// Step 3: Set the HttpDNS resolution proxy.
[TXLiveBase sharedInstance].delegate = self;
```

# 7. HEVC Adaptive Downgrade Play

The player supports playing links that contain both HEVC and other video encoding formats, such as H.264. When the player device does not support the HEVC format, it will automatically downgrade to playing videos in the configured alternative encoding format (such as H.264).

Note: Supported from player version 11.7 of the premium version.





```
#import <CoreMedia/CoreMedia.h> //Import header file
NSDictionary *dic = @{
    VOD_KEY_VIDEO_CODEC_TYPE:@(kCMVideoCodecType_HEVC), // Specify the original HE
    VOD_KEY_BACKUP_URL:@"${backupPlayUrl}"}; // Set the backup playback link addres
[_txVodPlayer setExtentOptionInfo:dic];
// Set the original HEVC playback link
[_txVodPlayer startVodPlay:@"${hevcPlayUrl}"];
```

# 8. Volume Normalization

The player supports automatically adjusting the volume when playing audio to ensure that the volume of all audio is consistent. This can avoid problems with some audio being too loud or too quiet, providing a better auditory experience. Use TXVodPlayer#setAudioNormalization to set the volume normalization, with a loudness range of -70 to 0 (LUFS), and custom values are also supported. Note: Supported from player version 11.7 of the premium version.

/\*\*

Can be set to preset values (related classes or files: Android: TXVodConstants; iO Off: AUDIO\_NORMALIZATION\_OFF

```
On: AUDIO_NORMALIZATION_STANDARD (standard)
    AUDIO_NORMALIZATION_LOW (low)
    AUDIO_NORMALIZATION_HIGH (high)
Custom values can be set: from low to high, range -70 to 0 LUFS
*/
[_txVodPlayer setAudioNormalization:AUDIO_NORMALIZATION_STANDARD]; //
[_txVodPlayer setAudioNormalization:AUDIO_NORMALIZATION_OFF];
```

# **Player Event Listening**

You can bind a TXVodPlayListener listener to the TXVodPlayer object to use onPlayEvent (event notification) and onNetStatus (status feedback) to sync information to your application.

# Event notification (onPlayEvent)

## **Playback events**

| Event ID                        | Code | Description  |
|---------------------------------|------|--|
| PLAY_EVT_PLAY_BEGIN             | 2004 | Video playback started.  |
| PLAY_EVT_PLAY_PROGRESS          | 2005 | Video playback progress. The current<br>playback progress, loading progress,<br>and total video duration will be notified<br>of. |
| PLAY_EVT_PLAY_LOADING           | 2007 | The video is being loaded. The<br>LOADING_END event will be<br>reported if video playback resumes.                               |
| PLAY_EVT_VOD_LOADING_END        | 2014 | Video loading ended, and video playback resumed.   |
| VOD_PLAY_EVT_SEEK_COMPLETE      | 2019 | Seeking was completed. The seeking feature is supported by v10.3 or later.   |
| VOD_PLAY_EVT_LOOP_ONCE_COMPLETE | 6001 | A round of loop was completed. The loop feature is supported by v10.8 or later.  |
| VOD_PLAY_EVT_HIT_CACHE          | 2002 | Cache hit event at startup (supported since version 11.2).   |
| VOD_PLAY_EVT_VIDEO_SEI          | 2030 | Received SEI frame event (supported  |

|  |      | from player version 11.6 of the premium version).                                  |
|--|------|--|
| VOD_PLAY_EVT_HEVC_DOWNGRADE_PLAYBACK     | 2031 | HEVC downgrade playback occurs (supported by the player's advanced version 12.0).  |
| VOD_PLAY_EVT_VOD_PLAY_FIRST_VIDEO_PACKET | 2017 | The player receives the first frame data packet event (supported by version 12.0). |

# SEI frame

SEI (Supplemental Enhancement Information) frames are a type of frame used to transmit additional information. The premium version of the player will parse the SEI frames in the video stream and provide callbacks through the VOD\_PLAY\_EVT\_VIDEO\_SEI` event.

Note: Supported from player version 11.6 of the premium version.





```
-(void) onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary*
    if (EvtID == VOD_PLAY_EVT_VIDEO_SEI) {
        int seiType = [param objectForKey:EVT_KEY_SEI_TYPE]; // the type of video
        int seiSize = [param objectForKey:EVT_KEY_SEI_SIZE]; // the data size of vi
        NSData *seiData = [param objectForKey:EVT_KEY_SEI_DATA]; // the byte array
    }
}
```

Warning events

You can ignore the following events, which are only used to notify you of some internal events of the SDK.

| Event ID                          | Code | Description  |
|-----------------------------------|------|--|
| PLAY_WARNING_VIDEO_DECODE_FAIL    | 2101 | Failed to decode the current video frame.  |
| PLAY_WARNING_AUDIO_DECODE_FAIL    | 2102 | Failed to decode the current audio frame.  |
| PLAY_WARNING_RECONNECT            | 2103 | The network was disconnected, and automatic reconnection was performed (the PLAY_ERR_NET_DISCONNECT event will be thrown after three failed attempts). |
| PLAY_WARNING_HW_ACCELERATION_FAIL | 2106 | Failed to start the hardware decoder, and the software decoder was used instead.   |

# **Connection events**

The following server connection events are mainly used to measure and collect the server connection time:

| Event ID                   | Code | Description   |  |  |
|----------------------------|------|---|--|--|
| PLAY_EVT_VOD_PLAY_PREPARED | 2013 | The player has been prepared and can start playback. If<br>autoPlay is set to false, you need to call<br>resume after receiving this event to start playback. |  |  |
| PLAY_EVT_RCV_FIRST_I_FRAME | 2003 | The network received the first renderable video data packet (IDR).  |  |  |

# Image quality events

The following events are used to get image change information:

| Event ID                   | Code | Description                   |
|----------------------------|------|-------------------------------|
| PLAY_EVT_CHANGE_RESOLUTION | 2009 | The video resolution changed. |
| PLAY_EVT_CHANGE_ROATION    | 2011 | The MP4 video was rotated.    |

# Video information events

| Event ID                   | Code | Description  |
|----------------------------|------|--|
| PLAY_EVT_GET_PLAYINFO_SUCC | 2010 | Obtained the information of the file played back successfully. |

If you play back a video through fileId and the playback request succeeds, the SDK will notify the upper layer of some request information, and you can parse param to get the video information after receiving the PLAY\_EVT\_GET\_PLAYINFO\_SUCC event.

| Video Information       | Description   |
|-------------------------|---|
| EVT_PLAY_COVER_URL      | Video thumbnail URL   |
| EVT_PLAY_URL            | Video playback URL  |
| EVT_PLAY_DURATION       | Video duration  |
| EVT_KEY_WATER_MARK_TEXT | Ghost watermark text content (supported from version 11.6). |





```
-(void) onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary*
{
    if (EvtID == PLAY_EVT_VOD_PLAY_PREPARED) {
        // The player preparation completion event is received, and you can call th
    } else if (EvtID == PLAY_EVT_PLAY_BEGIN) {
        // The playback start event is received
    } else if (EvtID == PLAY_EVT_PLAY_END) {
        // The playback end event is received
    }
}
```



# Ghost watermark

The content of the ghost watermark is filled in the player signature and is ultimately displayed on the playback end through collaboration between the cloud and the player, ensuring the security of the watermark throughout the transmission process. Follow the tutorial to configure the ghost watermark in the player signature. The content of the ghost watermark can be obtained through `[param objectForKey:@"EVT\_KEY\_WATER\_MARK\_TEXT"]` after receiving the `VOD\_PLAY\_EVT\_GET\_PLAYINFO\_SUCC` event from the player. For detailed usage tutorial, please refer to SuperPlayer Component > Ghost Watermark.

Note: Supported from player version 11.6.

## Playback error event

#### Note:

[-6004, -6010] Error events are supported since version 11.0.

| Event ID                          | Value | Meaning  |
|-----------------------------------|-------|--|
| PLAY_ERR_NET_DISCONNECT           | -2301 | Video data errors that cannot be recovered by<br>retrying the playback. For example, network<br>anomalies or download data errors that cause<br>demuxing timeouts or failures. |
| PLAY_ERR_HLS_KEY                  | -2305 | HLS decryption key retrieval failure.  |
| VOD_PLAY_ERR_SYSTEM_PLAY_FAIL     | -6004 | System player playback error.  |
| VOD_PLAY_ERR_DECODE_VIDEO_FAIL    | -6006 | Video decoding error or unsupported video format.  |
| VOD_PLAY_ERR_DECODE_AUDIO_FAIL    | -6007 | Audio decoding error or unsupported audio format.  |
| VOD_PLAY_ERR_DECODE_SUBTITLE_FAIL | -6008 | Subtitle decoding error.   |
| VOD_PLAY_ERR_RENDER_FAIL          | -6009 | Video rendering error.   |
| VOD_PLAY_ERR_PROCESS_VIDEO_FAIL   | -6010 | Video post-processing error.   |
| VOD_PLAY_ERR_GET_PLAYINFO_FAIL    | -2306 | Failed to obtain the on-demand file information. It is recommended to check whether the AppId, FileId or Psign is filled in correctly.   |

# Status feedback (onNetStatus)

The status feedback is triggered once every 0.5 seconds to provide real-time feedback on the current status of the pusher. It can act as a dashboard to inform you of what is happening inside the SDK so you can better understand the current video playback status.

| Parameter        | Description   |
|------------------|---|
| CPU_USAGE        | Current instantaneous CPU utilization   |
| VIDEO_WIDTH      | Video resolution - width  |
| VIDEO_HEIGHT     | Video resolution - height   |
| NET_SPEED        | Current network data reception speed in KBps  |
| VIDEO_FPS        | Current video frame rate of streaming media   |
| VIDEO_BITRATE    | Current video bitrate in bps of streaming media                                       |
| AUDIO_BITRATE    | Current audio bitrate in bps of streaming media                                       |
| V_SUM_CACHE_SIZE | Buffer (`jitterbuffer`) size. If the current buffer length is 0, lag will occur soon. |
| SERVER_IP        | Connected server IP   |

Below is the sample code of using `onNetStatus` to get the video playback information:




```
- (void) onNetStatus: (TXVodPlayer *)player withParam: (NSDictionary *)param {
    // Get the current CPU utilization
    float cpuUsage = [[param objectForKey:@"CPU_USAGE"] floatValue];
    // Get the video width
    int videoWidth = [[param objectForKey:@"VIDEO_WIDTH"] intValue];
    // Get the video height
    int videoHeight = [[param objectForKey:@"VIDEO_HEIGHT"] intValue];
    // Get the real-time speed
    int speed = [[param objectForKey:@"NET_SPEED"] intValue];
    // Get the current video frame rate of streaming media
    int fps = [[param objectForKey:@"VIDEO_FPS"] intValue];
```

```
// Get the current video bitrate in Kbps of streaming media
int videoBitRate = [[param objectForKey:@"VIDEO_BITRATE"] intValue];
// Get the current audio bitrate in Kbps of streaming media
int audioBitRate = [[param objectForKey:@"AUDIO_BITRATE"] intValue];
// Get the buffer (`jitterbuffer`) size. If the current buffer length is 0, lag
int jitterbuffer = [[param objectForKey:@"V_SUM_CACHE_SIZE"] intValue];
// Get the connected server IP
NSString *ip = [param objectForKey:@"SERVER_IP"];
}
```

### Other functions

### HLS live video source playback

The premium version of the player supports playing HLS live video sources. Starting from version 11.8, it supports live video sources with HLS EVENT. Usage is as follows:





```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setMediaType:MEDIA_TYPE_HLS_LIVE]; // Specify the HLS live media type
[_txVodPlayer setConfig:_config];
[_txVodPlayer startVodPlay:${YOUR_HSL_LIVE_URL}];
```

### Scenario-Specific Features

### 1. Dynamically Setting AudioSession

Sometimes it is necessary to dynamically set the audio output mode based on the scene, especially for iPhones, which natively support multiple audio playback and background modes. Therefore, we support the following three main modes based on the user's scenario:

AVAudioSessionCategoryPlayback: Exclusive playback in the background.

AVAudioSessionCategoryPlayAndRecord: Exclusive playback in the background.

AVAudioSessionCategoryAmbient: Mixed playback.

You can use the above modes to set the Category and Option of AudioSession according to the current scenario to achieve your purpose. The following are two examples of settings for different scenarios (the settings below can be dynamically adjusted and set according to your own scenario):

**Scenario 1**: Playlist scenario (video playback needs to support silent playback in the playlist and does not interrupt external audio playback).





[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPlayback withOpt
[[AVAudioSession sharedInstance] setActive:YES error:nil];

**Scenario 2: Playback details scenario** (the video details have sound, and temporarily interrupt external audio. After the video playback is complete, the external audio will be resumed).





[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryAmbient withOpti [[AVAudioSession sharedInstance] setActive:NO withOptions:AVAudioSessionSetActiveOp

#### 2. SDK-based demo component

Based on the Player SDK, Tencent Cloud has developed a player component. It integrates quality monitoring, video encryption, Top Speed Codec, definition selection, and small window playback and is suitable for all VOD and live playback scenarios. It encapsulates complete features and provides upper-layer UIs to help you quickly create a playback program comparable to popular video apps.

### 3. Open-source GitHub projects

Based on the Player SDK, Tencent Cloud has developed immersive video player, video feed stream, and multi-layer reuse components and will provide more user scenario-based components on future versions. You can download the Player for iOS to try out the components.

# Android Integration Integration Guide

Last updated : 2024-04-26 11:09:31

This document describes how to quickly integrate RT-Cube's player SDK into your project. Different versions of the SDK can be integrated in the same way.

### **Environment Requirements**

Android Studio 2.0 or above Android 4.1 (SDK API level 16) or above

### Integrating the SDK (AAR)

You can use Gradle to automatically load the AAR file or manually download the AAR file and import it into your project.

### Method 1: automatic loading (AAR)

The player SDK has been released to the mavenCentral repository, and you can configure it in Gradle to download LiteAVSDK\_Player\_Premium updates automatically.

Open your project with Android Studio and modify the build.gradle file as described below to complete the integration.

| Demo – build.gradle (:app)  |  |  |  |  |  |  |
|---|--|--|--|--|--|--|
| 🗌 🔲 Google Pixel 6 🔻 🕼 🚓 🛎 🖏 \land 🦚 義 📕 🛛 🏘 🖳 🏘 🖓 Git: 🖌 🗸 🧦 🔇 😏   |  |  |  |  |  |  |
|   |  |  |  |  |  |  |
| 💠 🗕 🎢 build.gradle (:app) 🛛 🍓 res/values/strings.xml 🛛 🍓 en/strings.xml 👋 🍓 en/strings.xml 🖉 🖨 TXTrackInfo              |  |  |  |  |  |  |
| ub/Player_Android/Demo e changed since last project sync. A project sync may be necessary for the IDE to work properly. |  |  |  |  |  |  |
| 10 targetSdkVersion rootProject.ext.targetSdkVersion  |  |  |  |  |  |  |
| 11 versionCode 24   |  |  |  |  |  |  |
| 12 versionName "11.3.0.3122"  |  |  |  |  |  |  |
|   |  |  |  |  |  |  |
| 14 multiDexEnabled true   |  |  |  |  |  |  |
| 15 🕞 ndk { NdkOptions it ->   |  |  |  |  |  |  |
| non] 16 [abiFilters "armeabi-v7a", "arm64-v8a"] 3   |  |  |  |  |  |  |
| 17 () }   |  |  |  |  |  |  |
| 18 🗇 }  |  |  |  |  |  |  |
|   |  |  |  |  |  |  |
| 20 🕀 signingConfigs {}  |  |  |  |  |  |  |
| 24  |  |  |  |  |  |  |
| 25 🕀 o buildTypes {}  |  |  |  |  |  |  |
| 32  |  |  |  |  |  |  |
| 33 <b>1</b>   |  |  |  |  |  |  |
|   |  |  |  |  |  |  |
| ZE N dependencies (   |  |  |  |  |  |  |
| 35 ▶   dependencies i   |  |  |  |  |  |  |
|   |  |  |  |  |  |  |
| 37 Implementation file/ree(dir: 'Libs', include: ['*.jar'])   |  |  |  |  |  |  |
| 38 implementation 'com.tencent.liteav:LiteAVSDK_Player_Premium:la   |  |  |  |  |  |  |
| <pre>39 implementation project(':deprecatedcommon')</pre>   |  |  |  |  |  |  |
| 40 implementation project(':superplayerdemo')   |  |  |  |  |  |  |
| 41 implementation proiect(':feedvideo') android{  |  |  |  |  |  |  |
| <pre>25</pre>   |  |  |  |  |  |  |

1. Add the mavenCentral repository to the build.gradle in your project's root directory.



```
repositories {
  mavenCentral()
}
```

2. Open the build.gradle in the app directory and add the LiteAVSDK\_Player dependencies to dependencies .





```
dependencies {
    // This configuration integrates the latest version of `LiteAVSDK_Player_Premium`
    implementation 'com.tencent.liteav:LiteAVSDK_Player_Premium:latest.release'
    // To integrate an earlier version such as 10.8.0.29000, configure as follows:
    // implementation 'com.tencent.liteav:LiteAVSDK_Player_Premium:10.8.0.29000'
}
```

3. In defaultConfig, specify the CPU architecture to be used by the application. Currently, LiteAVSDK\_Player supports armeabi, armeabi-v7a, and arm64-v8a.



```
defaultConfig {
  ndk {
     abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
  }
}
```

4. Click the Sync Now button

svnc the SDK If vo

to sync the SDK. If you have no problem accessing Maven Central, the SDK will be downloaded and integrated into your project automatically.

### Method 2: manual download (AAR)

If you have problem accessing Maven Central, you can manually download the SDK and integrate it into your project.

- 1. Download LiteAVSDK\_Player\_Premium and decompress the file.
- 2. Copy the AAR file in the SDK directory to the app/libs directory of your project.



3. Add flatDir to build.gradle under your project's root directory to specify the local path for the repository.



4. Add the LiteAVSDK\_Player\_Premium dependency and then add code that references the AAR file in

app/build.gradle .



|        |      |  |                     |          |              | Demo huild gradle (John)  |
|--------|------|--|---------------------|----------|--------------|---|
|        |      |  |                     |          |              |   |
| ►      |      | $S \leftarrow \rightarrow \checkmark \mid \checkmark app \checkmark$ | Google Pixel 6 🔻    |          |              | E 🕼 🗥 🏟 📕   🌌 🗓 Git: 🖌 🗸 🥕 🕓 つ  |
| De     | emo) | 👌 app 👌 🗬 build.gradle   |                     |          |              |   |
| ect    | 🔲 Pr | roject 🔫 🤤   | 🕀 🖻 😤 🗢 🖨           | <b>@</b> |              | e (:app) 🛛 🍓 res/values/strings.xml 🛛 🏭 en/strings.xml 👋 🛃 en/strings.xml 🖉 🧟 TXTrackl          |
| Proj   | ~ 📭  | <b>Demo</b> ~/av_project_code/Github/F                               |                     | Gra      | dle files ha | ave changed since last project sync. A project sync may be necessary for the IDE to work proper |
|        |      |  |                     |          |              |   |
| ŧ      |      | 🖿 .idea  |                     |          |              |   |
| mmo    |      | napp   |                     |          |              | multiDexEnabled true  |
| Ŭ<br>d |      | > build  |                     |          |              | ndk { NdkOptions it ->  |
|        |      |  | 40.0.0.00000        |          |              | abiFilters "armeabi-v7a", "arm64-v8a"   |
| sts    |      |  | um_10.9.0.29003.aar |          |              | ι   |
| anpe   |      |  |                     |          |              | , , , , , , , , , , , , , , , , , , ,   |
| III Re |      | w build.gradle   |                     |          |              | }   |
| Ъ<br>Г |      | 🗧 proguard-rules.pro   |                     |          |              |   |
|        |      | Common [deprecatedcommon]  |                     |          |              | <pre>signingConfigs {}</pre>  |
| ger    |      | 🖬 feedvideo  |                     |          |              |   |
| lana   |      | 🖿 gradle   |                     |          |              | huildTypes { }  |
| Se M   |      | 🖬 shortvideo   |                     |          |              | borculypes l  |
| ouro   |      | superplayerdemo  |                     |          |              |   |
| Res    |      | Superplayerkit   |                     |          |              |   |
| î      |      | Noacommon  |                     |          |              |   |
| ē      |      | aradle.properties  |                     |          | ▶ ⊟dep       | pendencies {  |
| Intor  |      | angradlew  |                     |          |              |   |
| Stru   |      | 🖞 gradlew.bat  |                     |          |              | implementation fileTrac(dim, lliket, include, [t, incl])  |
| •      |      | 📊 local.properties   |                     |          |              | implementation file/ree(dir: 'libs', include: ['*.jar'])  |
| es     |      | 🗬 settings.gradle  |                     | 38       |              | <pre>implementation(name:'LiteAVSDK_Player_Premium_10.9.0.29003',</pre>                         |
| vorit  | > 🏨  | External Libraries   |                     |          |              | <pre>implementation project(':deprecatedcommon')</pre>  |
| F Fa   | > 🚾  | Scratches and Consoles   |                     |          |              | <pre>implementation project(':superplayerdemo')</pre>   |
|        |      |  |                     |          |              | <pre>implementation project(':feedvideo')</pre>   |
| nts    |      |  |                     |          |              | implementation project(':chaptyideo')   |
| arial  |      |  |                     |          |              |   |
| V bli  |      |  |                     |          |              |   |
| ( Bu   |      |  |                     |          | depe         | pendencies{}  |
|        |      |  |                     |          |              |   |





implementation(name:'LiteAVSDK\_Player\_Premium\_10.9.0.29003', ext:'aar')

5. In defaultConfig of app/build.gradle, specify the CPU architecture to be used by the application. Currently, LiteAVSDK\_Player supports armeabi, armeabi-v7a, and arm64-v8a.



```
defaultConfig {
  ndk {
     abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
  }
}
```

6. Click **Sync Now** to complete the integration of LiteAVSDK.

### Integrating the SDK (JAR)

### 🕗 Tencent Cloud

If you do not want to import the AAR library, you can also integrate LiteAVSDK by importing JAR and SO libraries. 1. Download LiteAVSDK\_Player\_Premium and decompress it. Find LiteAVSDK\_Player\_Premium\_xxx.zip ( xxx is the version number) in the SDK directory. After decompression, you can get the libs directory, which contains the JAR file and folders of SO files as listed below:



If you also need the .so file for the armeabi architecture, copy the armeabi-v7a directory and rename it armeabi .

2. Copy the JAR file and armeabi, armeabi-v7a, and arm64-v8a folders to the app/libs directory.



3. Add code that references the JAR library in <code>app/build.gradle</code> .









```
dependencies {
  implementation fileTree(dir:'libs',include:['*.jar'])
}
```

4. Add **flatDir** to build.gradle under the project's root directory to specify the local path for the repository.

|   | LiteAVSDKDemo – build   |  |  |  |  |  |  |  |
|---|---|--|--|--|--|--|--|--|
| 🔄 🛱 🕄   $\leftarrow$ $\rightarrow$   $\checkmark$ 🔺 app 💌 No De   | evices 🔻 🕨 🖒 🖶 🏟 🖏 🕫 🖌 🏘 💷 🏘 💷 🔍  |  |  |  |  |  |  |  |
| LiteAVSDKDemo 👌 🗬 build.gradle  |   |  |  |  |  |  |  |  |
| tig 🔲 Project 👻 😌 😤 🗘 🗢   | earrow |  |  |  |  |  |  |  |
| LiteAVSDKDemo ~/AndroidStudioProj<br>.gradle<br>.idea<br>.grapp<br>b libs<br>b src<br>gitignore   | <pre>1 // Top-level build file where you can add configuration 2</pre>  |  |  |  |  |  |  |  |
| <ul> <li>Begin build.gradle</li> <li>Begin build.gradle</li> <li>Begin build.gradle</li> <li>Begin build.gradle</li> <li>Igradle</li> <li>Begin build.gradle</li> <li>Begin build.gradle</li> </ul>                   | 9<br>10 7 // NOTE: Do not place your application dependent<br>11 7 // in the individual module build.gradle files<br>12 4 }   |  |  |  |  |  |  |  |
| <ul> <li>build.gradle</li> <li>gradle.properties</li> <li>gradlew</li> <li>gradlew.bat</li> <li>local.properties</li> <li>settings.gradle</li> <li>IIII External Libraries</li> <li>Scratches and Consoles</li> </ul> | <pre>13</pre>   |  |  |  |  |  |  |  |





6. In the defaultConfig of app/build.gradle, specify the CPU architecture to be used by the application (currently, LiteAVSDK supports armeabi, armeabi-v7a, and arm64-v8a).



```
defaultConfig {
  ndk {
     abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
  }
}
```

7. Click **Sync Now** to complete the integration.

### **Configuring Permissions**



Configure permissions for your application in AndroidManifest.xml . LiteAVSDK needs the following permissions:



```
<!--network permission-->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<!--storage-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

#### Network security configuration allows the app to send HTTP requests

For security reasons, starting from Android P, Google requires that requests from apps use encrypted connections. The player SDK will start a localsever to proxy HTTP requests. If your app's targetSdkVersion is greater than or equal to 28, you can enable sending HTTP requests to 127.0.0.1 through network security configuration. Otherwise, an error 'java.io.IOException: Cleartext HTTP traffic to 127.0.0.1 not permitted' will occur during playback, causing video playback to fail. The configuration steps are as follows:

1. Create a res/xml/network\_security\_config.xml file in the project and set the network security configuration.



<?xml version="1.0" encoding="utf-8"?> <network-security-config>



2. Add the following attribute to the application tag in the AndroidManifest.xml file.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
<application android:networkSecurityConfig="@xml/network_security_config"
... >
```



```
</application> </manifest>
```

## **Configuring Obfuscation Rules**

In the proguard-rules.pro file, add LiteAVSDK classes to the "do not obfuscate" list.



-keep class com.tencent.\*\* { \*;}

### **Configuring License**

Enter VOD console. to apply for a trial license as instructed in Adding and Renewing a License. If no license is configured, video playback will fail. You will get two strings: a license URL and a decryption key. After obtaining the License information, you need to initialize and configure the License before calling the relevant interfaces of the SDK. For detailed tutorials, please see Configuring View License.

### FAQs

# What should I do if a duplicate symbol error occurs because my project integrates multiple editions of LiteAVSDK such as CSS, TRTC, and Player?

If you integrate two or more editions of LiteAVSDK (MLVB, Player, TRTC, UGSV), a library conflict error will occur when you build your project. This is because some symbol files are shared among the underlying libraries of the SDKs. To solve the problem, we recommend you integrate the All-in-One SDK, which includes the features of MLVB, Player, TRTC, and UGSV. For details, see SDK Download.

# **VOD** Scenario

Last updated : 2024-08-07 15:16:13

## Limits

1. To try out all features of the player, we recommend you activate VOD. If you don't have an account yet, sign up for one first. If you don't use the VOD service, you can skip this step; however, you will only be able to use basic player features after integration.

2. Download and install Android Studio. If you have already done so, skip this step.

### This Document Describes

How to integrate the Tencent Cloud Player SDK for Android.

How to use the Player SDK for VOD playback.

How to use the underlying capabilities of the Player SDK to implement more features.

### **SDK** Integration

#### Step 1. Integrate the SDK ZIP file

Download and integrate the SDK ZIP file as instructed in Integration Guide.

#### Step 2. Configure the license

If you have obtained a license, you can view the license URL and key in the VOD console.

If you don't have the required license yet, you can get it as instructed in Adding and Renewing a License.

After obtaining the License information, you need to initialize and configure the License before calling the relevant interfaces of the SDK. For detailed tutorials, please see Configuring View License.

#### Step 3. Add a view

The SDK provides TXCloudVideoView for video rendering by default. First, add the following code to the layout XML file:



```
<com.tencent.rtmp.ui.TXCloudVideoView
android:id="@+id/video_view"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_centerInParent="true"
android:visibility="gone"/>
```

#### Step 4. Create a player object



Create the **TXVodPlayer** object and use the setPlayerView API to associate the object with the **video\_view** control just added to the UI.



// `mPlayerView` is the video rendering view added in step 3
TXCloudVideoView mPlayerView = findViewById(R.id.video\_view);
// Create a player object
TXVodPlayer mVodPlayer = new TXVodPlayer(getActivity());
// Associate the player object with the video rendering view
mVodPlayer.setPlayerView(mPlayerView);



#### Step 5. Start playback

TXVodPlayer supports two playback modes for you to choose as needed:

Through URL

Through FileId

TXVodPlayer will internally recognize the playback protocol automatically. You only need to pass in your playback URL to the startVodPlay function.



// Play back a video resource at a URL
String url = "http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
mVodPlayer.startVodPlay(url);



```
// Play back a local video resource
String localFile = "/sdcard/video.mp4";
mVodPlayer.startVodPlay(localFile);
// Starting from version 11.8, the player supports content:// URI video resources a
// Play content:// URI video resources
String localFile = "content://xxx/xxx/video.mp4";
mVodPlayer.startVodPlay(localFile);
// Play asset catalog video resources, the passed address must start with asset://
```

// Play asset catalog video resources, the passed address must start with asset://
String localFile = "asset://video.mp4";
mVodPlayer.startVodPlay(localFile





```
// We recommend you use the following new API:
// `psign` is a player signature. For more information on the signature and how to
TXPlayInfoParams playInfoParam = new TXPlayInfoParams(1252463788, // `appId` of the
"4564972819220421305", // `fileId` of the video
"psignxxxxxxx"); // The player signature
mVodPlayer.startVodPlay(playInfoParam);
```

```
// Legacy API, which is not recommended
TXPlayerAuthBuilder authBuilder = new TXPlayerAuthBuilder();
authBuilder.setAppId(1252463788);
authBuilder.setFileId("4564972819220421305");
```



```
mVodPlayer.startVodPlay(authBuilder);
```

Find the target video file in Media Assets, and you can view the FileId below the filename. Play back the video through the FileId, and the player will request the backend for the real playback URL. If the network is abnormal or the FileId doesn't exist, the TXLiveConstants.PLAY\_ERR\_GET\_PLAYINFO\_FAIL event will be received; otherwise, TXLiveConstants.PLAY\_EVT\_GET\_PLAYINFO\_SUCC will be received, indicating that the request succeeded.

#### Step 6. Stop playback

**Remember to terminate the view control** when stopping the playback, especially before the next call of startVodPlay. This can prevent memory leak and screen flashing issues.

In addition, when exiting the playback UI, you need to call the <code>onDestroy()</code> function for the rendering view. This can prevent memory leak and "Receiver not registered" alarms.





```
@Override
public void onDestroy() {
    super.onDestroy();
    mVodPlayer.stopPlay(true); // `true` indicates to clear the last-frame image
    mPlayerView.onDestroy();
}
```

#### Note:

The boolean parameter of stopPlay indicates whether to clear the last-frame image. Early versions of the live player of the RTMP SDK don't have an official pause feature; therefore, this boolean value is used to clear the last-


frame image.

If you want to retain the last-frame image after VOD stops, simply do nothing after receiving the playback stop event; playback will stop at the last frame by default.

# **Basic Feature Usage**

## 1. Playback control

Starting playback





Video on Demand

```
// Start playback
mVodPlayer.startVodPlay(url)
```

#### Pausing playback



// Pause the video
mVodPlayer.pause();

**Resuming playback** 



// Resume the video
mVodPlayer.resume();

### Stopping playback



// Stop the video
mVodPlayer.stopPlay(true);

#### Adjusting playback progress (seek)

When the user drags the progress bar, seek can be called to start playback at the specified position. The Player SDK supports accurate seek.





int time = 600; // In seconds if the value is of `int` type
// float time = 600; // In seconds if the value is of `float` type
// Adjust the playback progress
mVodPlayer.seek(time);

#### Precise and Imprecise Seek

Starting from version 11.8 of the player SDK, it is supported to specify precise or imprecise seek when calling the seek interface.





```
float time = 600; // Unit is seconds for float type
// Adjust progress
mVodPlayer.seek(time, true); // Precise seek
mVodPlayer.seek(time, false); // Imprecise seek
```

#### Seek to the specified Program Date Time (PDT) point in the video stream

To seek to the specified Program Date Time (PDT) point in the video stream, which enables functions such as fastforward, rewind, and progress bar jumping, currently only HLS video format is supported. **Note:** 



Starting from version 11.6 of the player's premium edition, this function is supported.



long pdtTimeMs = 600; // Unit is milliseconds
mVodPlayer.seekToPdtTime(time);

#### Specifying playback start time

You can specify the playback start time before calling startVodPlay for the first time.





```
float startTimeInSecond = 60; // Unit: Second
mVodPlayer.setStartTime(startTimeInSecond); // Set the playback start time
mVodPlayer.startVodPlay(url);
```

#### 2. Image adjustment

#### view: size and position

You can modify the size and position of video images by adjusting the size and position of the video\_view control added in the Add a view step during SDK integration.

### setRenderMode: Aspect fill or aspect fit

| Value                         | Definition   |
|-------------------------------|--|
| RENDER_MODE_FULL_FILL_SCREEN  | Images are scaled to fill the entire screen, and the excess parts<br>are cropped. There are no black bars in this mode, but images<br>may not be displayed entirely.           |
| RENDER_MODE_ADJUST_RESOLUTION | Images are scaled so that the long side of the video fits the screen. Neither side exceeds the screen after scaling. Images are centered, and there may be black bars visible. |

### setRenderRotation: Image rotation

| Value                     | Definition   |
|---------------------------|--|
| RENDER_ROTATION_PORTRAIT  | Normal playback (the Home button is below the video image)   |
| RENDER_ROTATION_LANDSCAPE | Clockwise rotation of the image by 270 degrees (the Home button is on the left of the video image) |





// Fill the screen at the original aspect ratio
mVodPlayer.setRenderMode(TXLiveConstants.RENDER\_MODE\_FULL\_FILL\_SCREEN);
// Normal playback (the Home button is below the video image)
mVodPlayer.setRenderRotation(TXLiveConstants.RENDER\_ROTATION\_PORTRAIT);

### 3. Adjustable-Speed playback

The VOD player supports adjustable-speed playback. You can use the setRate API to set the VOD playback speed, such as 0.5x, 1.0x, 1.2x, and 2x speed.



// Set playback at 1.2X rate
mVodPlayer.setRate(1.2);

## 4. Playback loop



// Set playback loop
mVodPlayer.setLoop(true);
// Get the current playback loop status
mVodPlayer.isLoop();

## 5. Muting/Unmuting





// Mute or unmute the player. true: Mute; false: Unmute
mVodPlayer.setMute(true);

#### 6. Screencapturing

Call **snapshot** to take a screenshot of the current video frame. This method captures only the video frame. To capture the UI, use the corresponding API of the Android system.



```
// Take a screenshot
mVodPlayer.snapshot(new ITXSnapshotListener() {
    @Override
    public void onSnapshot(Bitmap bmp) {
        if (null != bmp) {
            // Get the screenshot bitmap
            }
        }
});
```



#### 7. Roll image ad

The Player SDK allows you to add roll images on the UI for advertising as follows:

If autoPlay is set to NO, the player will load the video normally but will not immediately start playing it back.

Users can see the roll image ad on the player UI after the player is loaded and before the video playback starts.

When the ad display stop conditions are met, the resume API will be called to start video playback.



```
mVodPlayer.setAutoPlay(false); // Set manual playback
mVodPlayer.startVodPlay(url); // The video will be loaded after `startVodPlay` i
// .....
// Display the ad on the player UI
```

## ठ Tencent Cloud

```
// .....
mVodPlayer.resume(); // Call `resume` to start playing back the video after the ad
```

#### 8. HTTP-REF

headers in TXVodPlayConfig can be used to set HTTP request headers, such as the Referer field commonly used to prevent the URL from being copied arbitrarily (Tencent Cloud provides a more secure signaturebased hotlink protection solution) and the Cookie field for client authentication



TXVodPlayConfig mPlayConfig = new TXVodPlayConfig(); Map<String, String> headers = new HashMap<>();

```
headers.put("Referer", "${Refer Content}");
mPlayConfig.setHeaders(headers);
mVodPlayer.setConfig(mPlayConfig);
```

#### 9. Hardware acceleration

It is extremely difficult to play back videos of the Blu-ray (1080p) or higher image quality smoothly if only software decoding is used. Therefore, if your main scenario is game live streaming, we recommend you use hardware acceleration.

Before switching between software and hardware decoding, you need to call **stopPlay** first. After the switch, you need to call **startVodPlay**; otherwise, severe blurs will occur.





mVodPlayer.stopPlay(true); mVodPlayer.enableHardwareDecode(true); mVodPlayer.startVodPlay(flvUrl, type);

#### 10. Definition settings

The SDK supports the multi-bitrate format of HLS, so users can switch between streams at different bitrates to switch the video definition. You can set the definition as follows:





// Get the array of multiple bitrates. The TXBitrateItem class fields mean: index ArrayList<TXBitrateItem> bitrates = mVodPlayer.getSupportedBitrates();
int index = bitrates.get(i).index; // Specify the bitrate index to be played
mVodPlayer.setBitrateIndex(index); // Switch the bitrate to the desired clarity

// Get the index of the currently played bitrate. The return value of -1000 is the
int index = mVodPlayer.getBitrateIndex();

During playback, you can call mVodPlayer.setBitrateIndex(int) at any time to switch the bitrate. During the switch, the data of another stream will be pulled. The SDK is optimized for Tencent Cloud multi-bitrate files to



implement smooth switching.

If you know the resolution information of the video stream in advance, you can specify the video resolution to be played before starting playback to avoid switching streams after playback. For detailed methods, refer to Player Configuration#Specifying Resolution Before Playback.

#### 11. Adaptive bitrate streaming

The SDK supports adaptive bitrate streaming of HLS. After this capability is enabled, the player can dynamically select the most appropriate bitrate for playback based on the current bandwidth. You can enable adaptive bitrate streaming as follows:



```
mVodPlayer.setBitrateIndex(-1); // Pass in `-1` for the `index` parameter
```

During playback, you can call mVodPlayer.setBitrateIndex(int) at any time to switch to another bitrate. After the switch, adaptive bitrate streaming will be disabled.

## 12. Enabling smooth bitrate switch

Before starting playback, you can enable smooth bitrate switch to seamlessly switch between different definitions (bitrates) during playback. If smooth bitrate switch is enabled, the transition between different bitrates will be smoother but will be more time-consuming. Therefore, this feature can be configured as needed.



```
TXVodPlayConfig mPlayConfig = new TXVodPlayConfig();
// If it is set to `true`, the bitrate can be switched smoothly when IDR frames are
mPlayConfig.setSmoothSwitchBitrate(true);
mVodPlayer.setConfig(mPlayConfig);
```

#### 13. Playback progress listening

There are two metrics for the VOD progress: **loading progress** and **playback progress**. Currently, the SDK notifies the two progress metrics in real time through event notifications. For more information on the event notification content, see Event Listening.

You can bind a **TXVodPlayerListener** listener to the **TXVodPlayer** object, and the progress notification will be called back to your application through the **PLAY\_EVT\_PLAY\_PROGRESS** event. The event information contains the above two progress metrics.





```
mVodPlayer.setVodListener(new ITXVodPlayListener() {
    @Override
    public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
        if (event == TXLiveConstants.PLAY_EVT_PLAY_PROGRESS) {
            // Loading progress in ms
            int playable_duration_ms = param.getInt(TXLiveConstants.EVT_PLAYABLE_DU
            mLoadBar.setProgress(playable_duration_ms); // Set the loading progress
            // Playback progress in ms
            int progress_ms = param.getInt(TXLiveConstants.EVT_PLAY_PROGRESS_MS);
            mSeekBar.setProgress(progress_ms); // Set the playback progress bar
```

```
// Total video duration in ms
int duration_ms = param.getInt(TXLiveConstants.EVT_PLAY_DURATION_MS);
// It can be used to set duration display, etc.
// Get the PDT time. This function is supported starting from version 1
long pdt_time_ms = param.getLong(TXVodConstants.EVT_PLAY_PDT_TIME_MS);
}
}
@Override
public void onNetStatus(TXVodPlayer player, Bundle bundle) {
}
```

## 14. Playback network speed listening

You can display the current network speed when the video is lagging by listening on events.

You can use the NET\_STATUS\_NET\_SPEED of onNetStatus to get the current network speed. For detailed directions, see Playback status feedback (onNetStatus).

After the PLAY\_EVT\_PLAY\_LOADING event is detected, the current network speed will be displayed.

After the PLAY\_EVT\_VOD\_LOADING\_END event is received, the view showing the current network speed will be hidden.





```
mVodPlayer.setVodListener(new ITXVodPlayListener() {
    @Override
    public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
        if (event == TXLiveConstants.PLAY_EVT_PLAY_LOADING) {
            // Show the current network speed
        }
        else if (event == TXLiveConstants.PLAY_EVT_VOD_LOADING_END) {
            // Hide the view showing the current network speed
        }
    }
}
```

```
@Override
public void onNetStatus(TXVodPlayer player, Bundle bundle) {
    // Get the real-time speed in Kbps
    int speed = bundle.getInt(TXLiveConstants.NET_STATUS_NET_SPEED);
});
```

## 15. Video resolution acquisition

The Player SDK plays back a video through a URL string. The URL doesn't contain the video information, and you need to access the cloud server to load such information. Therefore, the SDK can only send the video information to your application as event notifications. For more information, see Event Listening.

#### You can get the resolution information in the following two methods:

Method 1: Use the NET\_STATUS\_VIDEO\_WIDTH and NET\_STATUS\_VIDEO\_HEIGHT of onNetStatus to get the video width and height. For detailed directions, see Playback status feedback (onNetStatus). Method 2: Directly call TXVodPlayer.getWidth() and TXVodPlayer.getHeight() to get the current video width and height after receiving the PLAY\_EVT\_VOD\_PLAY\_PREPARED event callback from the player.





```
mVodPlayer.setVodListener(new ITXVodPlayListener() {
   @Override
   public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
   }
   @Override
   public void onNetStatus(TXVodPlayer player, Bundle bundle) {
        // Get the video width
        int videoWidth = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_WIDTH);
        // Get the video height
        int videoHeight = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_HEIGHT);
```

```
});
// Get the video width and height. The values can be returned only after the `PLAY_
mVodPlayer.getWidth();
mVodPlayer.getHeight();
```

#### 16. Player buffer size

🕗 Tencent Cloud

During normal video playback, you can control the maximum size of the data buffered from the network in advance. If the maximum buffer size is not configured, the player will use the default buffer policy to guarantee a smooth playback experience.





```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setMaxBufferSize(10); // Maximum buffer size during playback in MB
mVodPlayer.setConfig(config); // Pass in `config` to `mVodPlayer`
```

#### 17. Local video cache

In short video playback scenarios, the local video file cache is required, so that general users don't need to consume traffic again to reload an already watched video.

Supported format: The SDK supports caching videos in two common VOD formats: HLS (M3U8) and MP4.

**Enablement time:** The SDK doesn't enable the caching feature by default. We recommend you do not enable it for scenarios in which most videos are watched only once.

**Enablement method:** This feature can be enabled in the player and takes effect globally. To enable it, you need to configure two parameters: local cache directory and cache size.



```
File sdcardDir = getApplicationContext().getExternalFilesDir(null);
if (sdcardDir != null) {
    // Set the global cache directory of the playback engine
    TXPlayerGlobalSetting.setCacheFolderPath(sdcardDir.getPath() + "/txcache");
    // Set the global cache directory and cache size in MB of the playback engine
    TXPlayerGlobalSetting.setMaxCacheSize(200);
```

// Use the player

#### Note:

The TXVodPlayConfig#setMaxCacheItems API used for configuration on earlier versions has been deprecated and is not recommended.

#### 18. DRM-encrypted video playback

#### Note:

This feature requires the premium version of the player to be supported.

The premium version of Player SDK supports playback of commercial DRM-encrypted videos, currently supporting two DRM schemes: WideVine and Fairplay. For more information on commercial DRM, please refer to the product introduction.

Note: For DRM playback, please configure it through TXVodPlayer#setSurface before playback, otherwise playback exceptions may occur.

There are two ways to play DRM-encrypted videos:

Playback via Fileld

Customized playback configuration





// For DRM playback, please configure it through TXVodPlayer#setSurface before play
mVodPlayer.setSurface(mPlayerView.getHolder().getSurface());

```
// psign is the signature of the player. For information on generating and using pl
TXPlayInfoParams playInfoParam = new TXPlayInfoParams(${appId}, // Tencent Cloud ac
    ${fieId}, // The fileId of the DRM-encrypted video
    ${psgin}); // The player signature of the encrypted video
    mVodPlayer.startVodPlay(playInfoParam);
```

## ठ Tencent Cloud

Playback via FileId is suitable for integration with the VOD backend. This method is no different from playing regular FileId files, but DRM resources need to be configured in the VOD backend first, and the SDK will recognize and process them internally.



// For DRM playback, please configure it through TXVodPlayer#setSurface before play
mVodPlayer.setSurface(mPlayerView.getHolder().getSurface());

// Step 1: Set the DRM certificate provider environment. This step is not required // The Google DRM certificate provider environment defaults to the googleapis.com d TXPlayerGlobalSetting.setDrmProvisionEnv(TXPlayerGlobalSetting.DrmProvisionEnv.DRM\_



```
// Step 2: Play using the TXVodPlayer#startPlayDrm interface.
TXPlayerDrmBuilder builder = new TXPlayerDrmBuilder();
builder.setPlayUrl(${url}); // Set the URL of the video to be played.
builder.setKeyLicenseUrl(${keyLicneseUrl}); // Set decryption key URL
mVodPlayer.startPlayDrm(builder);
```

#### **19. External Subtitles**

#### Note:

This feature requires the premium version of the player to be supported.

The premium version of the player SDK supports adding and switching external subtitles, and now supports two subtitle formats: SRT and VTT.

Best practice: It is recommended to add subtitles and configure subtitle styles before calling startVodPlay. After receiving the VOD\_PLAY\_EVT\_VOD\_PLAY\_PREPARED event, call selectTrack to choose the subtitle. Adding subtitles does not automatically load them. After calling selectTrack, the subtitles will be loaded. The successful selection of subtitles will trigger the VOD\_PLAY\_EVT\_SELECT\_TRACK\_COMPLETE event callback. The usage is as follows:

Step 1: Set the subtitle rendering target object View.





#### Step 2: Add external subtitles.





// Pass in the subtitle URL, subtitle name, and subtitle type. It is recommended to
mVodPlayer.addSubtitleSource("https://mediacloud-76607.gzc.vod.tencent-cloud.com/De

Step 3: Switch subtitles after playback starts.




```
// After starting to play the video, select the added external subtitle. Please cal
@Override
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
    if (event == TXVodConstants.VOD_PLAY_EVT_VOD_PLAY_PREPARED) {
      List<TXTrackInfo> subtitleTrackInfoList = mVodPlayer.getSubtitleTrackInfo()
      for (TXTrackInfo track : subtitleTrackInfoList) {
        Log.d(TAG, "TrackIndex= " + track.getTrackIndex() + " ,name= " + track.
        if (TextUtils.equals(track.getName(), "subtitleName")) {
            // Selected Subtitles
            mVodPlayer.selectTrack(track.trackIndex);
        } else {
```



```
// If other subtitles are not required, proceed with deselectTrack.
                mVodPlayer.deselectTrack(track.trackIndex);
            }
        }
    }
}
// If needed, you can listen for track switching messages.
mVodPlayer.setVodListener(new ITXVodPlayListener() {
    @Override
    public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
        if (event == TXVodConstants.VOD_PLAY_EVT_SELECT_TRACK_COMPLETE) {
            int trackIndex = param.getInt(TXVodConstants.EVT_KEY_SELECT_TRACK_INDEX
            int errorCode = param.getInt(TXVodConstants.EVT_KEY_SELECT_TRACK_ERROR_
            Log.d(TAG, "receive VOD_PLAY_EVT_SELECT_TRACK_COMPLETE, trackIndex=" +
        }
    }
    @Override
    public void onNetStatus(TXVodPlayer player, Bundle status) {
    }
});
```

## Step 4: Configure subtitle style.

Subtitle style can be configured before or during playback.





// For detailed parameter configuration, please refer to the API documentation.
TXSubtitleRenderModel model = new TXSubtitleRenderModel();
model.canvasWidth = 1920; // Width of the subtitle rendering canvas
model.canvasHeight = 1080; // Height of the subtitle rendering canvas model
model.fontColor = 0xFFFFFF; // Set the font color of the subtitle, default is whi
model.isBondFontStyle = false; // Set whether the subtitle font is bold
mVodPlayer.setSubtitleStyle(model);

## 20. Switching between multiple audio tracks



## Note:

This feature requires the premium version of the player to be supported.

The advanced version of the player SDK supports switching between multiple audio tracks embedded in the video. Usage is as follows:



```
// Returns a list of audio track information
List<TXTrackInfo> soundTrackInfoList = mVodPlayer.getAudioTrackInfo();
for (TXTrackInfo trackInfo : soundTrackInfoList) {
    if (trackInfo.trackIndex == 0) {
        // Switch to the required audio track by checking the trackIndex or name
        mVodPlayer.selectTrack(trackInfo.trackIndex);
```

```
} else {
    // Deselect the unwanted audio track
    mVodPlayer.deselectTrack(trackInfo.trackIndex);
}
```

# Using Advanced Features

## 1. Video preloading

## Step 1. Use video preloading

In UGSV playback scenarios, the preloading feature contributes to a smooth viewing experience: When a video is being played, you can load the next video to be played back on the backend. When a user switches to the next video, it will already be loaded and can be played back immediately.

Video preloading can deliver an instant playback effect but has certain performance overheads. If your business needs to preload many videos, we recommend you use this feature together with video predownloading. This is how seamless switch works in video playback. You can use setAutoPlay in TXVodPlayer to implement the feature as follows:







```
// Play back video A: If `autoPlay` is set to `true`, the video will be immediately
String urlA = "http://1252463788.vod2.myqcloud.com/xxxxx/v.f10.mp4";
playerA.setAutoPlay(true);
playerA.startVodPlay(urlA);
```

```
// To preload video B when playing back video A, set `setAutoPlay` to `false`
String urlB = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
playerB.setAutoPlay(false);
playerB.startVodPlay(urlB); // The video won't be played back immediately but will
```



After video A ends and video B is automatically or manually switched to, you can call the resume function to immediately play back video B.

#### Note:

After autoPlay is set to false, make sure that video B has been prepared before calling resume, that is, you should call it only after the PLAY\_EVT\_VOD\_PLAY\_PREPARED event of video B (2013: the player has been prepared, and the video can be played back) is detected.



public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
 // When video A ends, directly start playing back video B for seamless switch
 if (event == PLAY\_EVT\_PLAY\_END) {

Video on Demand

## S Tencent Cloud

```
playerA.stop();
    playerB.setPlayerView(mPlayerView);
    playerB.resume();
}
```

## Step 2. Configure the video preloading buffer

You can set a large buffer to play back videos more smoothly under unstable network conditions.

You can set a smaller buffer to reduce the traffic consumption.

### Preloading buffer size

This API is used to control the maximum buffer size before the playback starts in preloading scenarios (that is, AutoPlay of the player is set to false before video playback starts).





```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setMaxPreloadSize(2); // Maximum preloading buffer size in MB. Set it based
mVodPlayer.setConfig(config); // Pass in `config` to `mVodPlayer`
```

#### Playback buffer size

During normal video playback, you can control the maximum size of the data buffered from the network in advance. If the maximum buffer size is not configured, the player will use the default buffer policy to guarantee a smooth playback experience.





```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setMaxBufferSize(10); // Maximum buffer size during playback in MB
mVodPlayer.setConfig(config); // Pass in `config` to `mVodPlayer`
```

## 2. Video predownloading

You can download part of the video content in advance without creating a player instance, so as to start playing back the video faster when using the player. This helps deliver a better playback experience. Before using the playback service, make sure that video cache has been set.



### Note:

1. TXPlayerGlobalSetting is the global cache setting API, and the original TXVodConfig API has been deprecated.

2. The global cache directory and size settings have a higher priority than those configured in TXVodConfig of the player.

Download via media URL

Download via media FileId

The code example for pre-downloading video via media URL is as follows:



 $//\ Set$  the global cache directory and cache size of the playback engine

## 🔗 Tencent Cloud

```
File sdcardDir = getApplicationContext().getExternalFilesDir(null);
// Set the global cache directory and cache size of the playback engine
if (sdcardDir != null) {
    TXPlayerGlobalSetting.setCacheFolderPath(sdcardDir.getPath() + "/PlayerCache");
    TXPlayerGlobalSetting.setMaxCacheSize(200); // Unit: MB
}
String palyrl = "http://****";
// Start predownloading
final TXVodPreloadManager downloadManager = TXVodPreloadManager.getInstance(getAppl
final int taskID = downloadManager.startPreload(playUrl, 3, 1920*1080, new ITXVodPr
    @Override
    public void onComplete(int taskID, String url) {
        Log.d(TAG, "preload: onComplete: url: " + url);
    }
    @Override
    public void onError(int taskID, String url, int code, String msg) {
        Log.d(TAG, "preload: onError: url: " + url + ", code: " + code + ", msg: "
    }
});
// Cancel predownloading
downloadManager.stopPreload(taskID);
```

#### Noe:

Pre-downloading via fileId is supported from version 11.3 onwards.

Pre-downloading via fileId is a time-consuming operation. Please do not call it on the main thread, otherwise an illegal call exception will be thrown. The preferredResolution parameter passed in when calling startPreload should be consistent with the preferred resolution set when starting playback, otherwise the expected effect will not be achieved. Here is an example of how to use it:





```
// Set the global cache directory and cache size for the playback engine File
File sdcardDir = getApplicationContext().getExternalFilesDir(null);
// Set the global cache directory and cache size for the playback engine
if (sdcardDir != null) {
    TXPlayerGlobalSetting.setCacheFolderPath(sdcardDir.getPath() + "/PlayerCache");
    TXPlayerGlobalSetting.setMaxCacheSize(200); // Unit: MB
}
// Start preloading
Runnable task = new Runnable() {
    @Override
```



```
public void run() {
        TXPlayInfoParams playInfoParams = new TXPlayInfoParams(${appId}, "${fileId}
                "${psign}");
        // Note: This is a time-consuming operation, do not call it on the main th
        mPreLoadManager.startPreload(playInfoParams, 3, 1920 * 1080, new ITXVodFile
            @Override
            public void onStart (int taskID, String fileId, String url, Bundle bundl
                // The onStart method will be called when the file URL is successfu
                Loq.d(TAG, "preload: onStart: taskID: " + taskID + ", fileId: " + f
            }
            Override
            public void onComplete(int taskID, String url) {
                Log.d(TAG, "preload: onComplete: url: " + url);
            }
            Override
            public void onError(int taskID, String url, int code, String msg) {
                Log.d(TAG, "preload: onError: url: " + url + ", code: " + code + ",
            }
        });
    }
};
new Thread(task).start();
// Cancel preloading
downloadManager.stopPreload(taskID);
```

## 3. Video download

Video download allows users to download online videos and watch them offline. If the video is encrypted, the downloaded video through the player SDK will be kept in an encrypted state locally and can only be decrypted and played through Tencent Cloud Player SDK. This can effectively prevent illegal dissemination of downloaded videos and protect video security.

As HLS streaming media cannot be directly saved locally, you cannot download them and play back them as local files. You can use the video download scheme based on TXVodDownloadManager to implement offline HLS playback.

#### Note:

Currently, TXVodDownloadManager does not support caching MP4 and FLV format filess. The Player SDK already supports playing back local MP4 and FLV files.

#### Step 1. Make preparations



When initializing the SDK, set the global storage path for video downloading, preloading, caching, and other functions. Usage is as follows:



```
File sdcardDir = context.getExternalFilesDir(null);
TXPlayerGlobalSetting.setCacheFolderPath(sdcardDir.getPath() + "/txcache");
```

TXVodDownloadManager is designed as a singleton; therefore, you cannot create multiple download objects. It is used as follows:





TXVodDownloadManager downloader = TXVodDownloadManager.getInstance();

#### Step 2. Start the download

You can start the download through Fileid or URL as follows:

Through Fileid

Through URL

For download throughFileid, you need to pass inAppID,Fileid, andqualityIdat least. For signedvideos, you also need to pass inpSign. If no specific value is passed in touserName, it will bedefaultby



default.

Note: You can download encrypted videos only through Fileid and must enter the psign parameter.



| // | QUALITY_2 | 40P | 240p |
|----|-----------|-----|------|
| // | QUALITY_3 | 60P | 360P |
| // | QUALITY_4 | 80P | 480p |
| // | QUALITY_5 | 40P | 540p |
| 11 |           |     | 700  |

- // QUALITY\_720P 720p
- // QUALITY\_1080P 1080p
- // QUALITY\_2K 2k



// QUALITY\_4K 4k
// The quality parameter can be customized to take the minimum value of the resolut
// The player SDK will select a stream that is less than or equal to the passed-in
TXVodDownloadDataSource source = new TXVodDownloadDataSource(1252463788, "456497281
downloader.startDownload(source)

You need to pass in the download URL at least. Only the non-nested HLS, i.e., single-bitstream HLS, is supported. If no specific value is passed in to userName, it will be default by default.



downloader.startDownloadUrl("http://1500005830.vod2.myqcloud.com/43843ec0vodtranscq



## Step 3. Receive the task information

Before receiving the task information, you need to set the callback listener first.



#### downloader.setListener(this);

You may receive the following task callbacks:

| Callback Message                               | Description  |
|--|--|
| void<br>onDownloadStart(TXVodDownloadMediaInfo | The task started, that is, the SDK started the download. |



| mediaInfo)   |  |
|--|--|
| void<br>onDownloadProgress(TXVodDownloadMediaInfo<br>mediaInfo)                        | Task progress. During download, the SDK will frequently call back this API. You can use mediaInfo.getProgress() to get the current progress.   |
| void<br>onDownloadStop(TXVodDownloadMediaInfo<br>mediaInfo)                            | The task stopped. When you call stopDownload to stop the download, if this message is received, the download is stopped successfully.  |
| void<br>onDownloadFinish(TXVodDownloadMediaInfo<br>mediaInfo)                          | Download was completed. If this callback is received, the entire file has been downloaded, and the downloaded file can be played back by TXVodPlayer .                               |
| void<br>onDownloadError(TXVodDownloadMediaInfo<br>mediaInfo, int error, String reason) | A download error occurred. If the network is disconnected during download, this API will be called back and the download task will stop. The error code is in TXVodDownloadManager . |

## The download error codes are as follows:

| Error Code             | Value | Description  |
|------------------------|-------|--|
| DOWNLOAD_SUCCESS       | 0     | Download succeeded.  |
| DOWNLOAD_AUTH_FAILED   | -5001 | Request for video information from the VOD console failed. Check whether the fileId and psign parameters are correct.  |
| DOWNLOAD_NO_FILE       | -5003 | The file of the specified definition does not exist.   |
| DOWNLOAD_FORMAT_ERROR  | -5004 | The downloaded file format is not supported.   |
| DOWNLOAD_DISCONNECT    | -5005 | Network disconnected. Check whether the network is normal.   |
| DOWNLOAD_HLS_KEY_ERROR | -5006 | Failed to obtain the HLS decryption key.   |
| DOWNLOAD_PATH_ERROR    | -5007 | Failed to access the download directory.<br>Check whether you have the permission to<br>access the download directory. |
| DOWNLOAD_403FORBIDDEN  | -5008 | Authentication information failed to pass<br>when downloading. Check whether the<br>signature (psign) has expired.     |

As the downloader can download multiple files at a time, the callback API carries the

 TXVodDownloadMediaInfo
 object. You can access the URL or
 dataSource
 to determine the download

 source and get other information such as download progress and file size.

### Step 4. Stop the download

You can call the downloader.stopDownload() method to stop the download. The parameter is the object returned by downloader.startDownload() . The SDK supports checkpoint restart. If the download directory is not changed, when you resume downloading a file, the download will start from the point where it stopped.

#### Step 5. Manage downloads

You can get the download lists of all accounts or the specified account.





```
// Get the download lists of all users
// You can distinguish between the download lists of different users by `userName`
// getDownloadMediaInfoList is a time-consuming function. Please do not call it in
List<TXVodDownloadMediaInfo> downloadInfoList = downloader.getDownloadMediaInfoList
if (downloadInfoList == null || downloadInfoList.size() <= 0) return;
// Get the download list of the `default` user
List<TXVodDownloadMediaInfo> defaultUserDownloadList = new ArrayList<>();
for(TXVodDownloadMediaInfo downloadMediaInfo : downloadInfoList) {
    if ("default".equals(downloadMediaInfo.getUserName())) {
        defaultUserDownloadList.add(downloadMediaInfo);
    }
```



#### }

To get the download information of a Fileid, such as the current download status and progress, you need to pass in AppID, Fileid, and qualityId.



// Get the download information of a `fileId`
TXVodDownloadMediaInfo downloadInfo = downloader.getDownloadMediaInfo(1252463788, "
// Get the total size of the file being downloaded in bytes. This API takes effect
// Note: The total size refers to the size of the original file uploaded to the VOD
int size = downloadInfo.getSize(); // Get the total size of the file being downloa
int duration = downloadInfo.getDuration(); // Get the total duration



```
int playableDuration = downloadInfo.getPlayableDuration(); // Get the playable dura
float progress = downloadInfo.getProgress(); // Get the download progress
String playPath = downloadInfo.getPlayPath(); // Get the offline playback path, whi
int downloadState = downloadInfo.getDownloadState(); // Get the download status. Fo
boolean isDownloadFinished = downloadInfo.isDownloadFinished(); // If `true` is ret
```

To get the download information of a URL, you need to pass in the URL information.



// Get the download information of a URL
TXVodDownloadMediaInfo downloadInfo = downloader.getDownloadMediaInfo("http://12531



To delete the download information and relevant file, you need to pass in the TXVodDownloadMediaInfo parameter.



// Delete the download information
boolean deleteRst = downloader.deleteDownloadMediaInfo(downloadInfo);

#### Step 6: Offline Playback after Download

The downloaded video supports playback without network connection. After downloading, you can obtain the download URL through TXVodDownloadMediaInfo#getPlayPath for playback.





// `getDownloadMediaInfoList` is a time-consuming function. Please do not call it i
List<TXVodDownloadMediaInfo> mediaInfoList = TXVodDownloadManager.getInstance().get
// Business logic finds the media object that needs to be played according to actua
for (TXVodDownloadMediaInfo mediaInfo : mediaInfoList) {

```
if (mediaInfo.getDownloadState() == TXVodDownloadMediaInfo.STATE_FINISH) { //
    mVodPlayer.startVodPlay(mediaInfo.getPlayPath()); // Play the downloaded v
}
```

```
}
```

## 4. Encrypted playback

The video encryption solution is used in scenarios where the video copyright needs to be protected, such as online education. To encrypt your video resources, you need to alter the player and encrypt and transcode video sources. For more information, see Media Encryption and Copyright Protection Overview.

After you get the appId as well as the encrypted video's fileId and psign in the Tencent Cloud console, you can play back the video as follows:



// `psign` is a player signature. For more information on the signature and how to
TXPlayInfoParams playInfoParam = new TXPlayInfoParams(1252463788, // `appId` of the
 "4564972819220421305", // `fileId` of the video
 "psignxxxxxx"); // The player signature

mVodPlayer.startVodPlay(playInfoParam);

## 5. Player configuration

Before calling statPlay, you can call setConfig to configure the player parameters, such as player connection timeout period, progress callback interval, and maximum number of cached files. TXVodPlayConfig allows you to configure detailed parameters. For more information, see Basic Configuration API. Below is the configuration sample code:



TXVodPlayConfig config = new TXVodPlayConfig();

config.setEnableAccurateSeek(true); // Set whether to enable accurate seek. Defaul config.setMaxCacheItems(5); // Set the maximum number of cached files to 5 config.setProgressInterval(200); // Set the progress callback interval in ms config.setMaxBufferSize(50); // Set the maximum preloading buffer size in MB mVodPlayer.setConfig(config); // Pass in `config` to `mVodPlayer`

#### Specifying resolution when playback starts

When playing back an HLS multi-bitrate video source, if you know the video stream resolution information in advance, you can specify the preferred resolution before playback starts, and the player will select and play back the stream at or below the preferred resolution. In this way, after playback starts, you don't need to call <code>setBitrateIndex</code> to switch to the required bitstream.





TXVodPlayConfig config = new TXVodPlayConfig();
// The parameter passed in is the product of the video width and height. You can pa
config.setPreferredResolution(TXLiveConstants.VIDEO\_RESOLUTION\_720X1280);
mVodPlayer.setConfig(config); // Pass in `config` to `mVodPlayer`

#### Specify media type before broadcasting

When the type of media asset to be played is known in advance, the playback speed can be enhanced by configuring

TXVodPlayConfig#setMediaType to reduce the internal playback type detection of the player SDK.

#### Note:

TXVodPlayConfig#setMediaType is supported starting from version 11.2.





```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setMediaType(TXVodConstants.MEDIA_TYPE_FILE_VOD); // Used to improve MP4 pl
// config.setMediaType(TXVodConstants.MEDIA_TYPE_HLS_VOD); // Used to improve HLS
mVodPlayer.setConfig(config);
```

Setting playback progress callback interval





TXVodPlayConfig config = new TXVodPlayConfig(); config.setProgressInterval(200); // Set the progress callback interval in ms mVodPlayer.setConfig(config); // Pass in `config` to `mVodPlayer`

## 6. HttpDNS resolution service

Mobile analysis (HTTPDNS) sends domain name resolution requests to DNS servers based on the HTTP protocol, replacing the traditional method of sending resolution requests to the operator's local DNS based on the DNS protocol.

This can avoid domain name hijacking and cross-network access issues caused by local DNS, and solve the problem of video playback failure caused by abnormal domain name resolution in mobile Internet services.

## Note:

HttpDNS resolution service is supported starting from version 10.9.

1. Enabling HTTPDNS resolution service

You can choose Tencent Cloud or other cloud providers to enable HTTPDNS resolution service. Make sure to integrate it into the player SDK after successful activation.

2. Accessing HTTPDNS resolution service in the player SD

Taking Tencent Cloud HTTPDNS as an example, the following shows how to access it in the player SDK: :



```
// Step 1: Turn on the HttpDNS resolution switch
TXLiveBase.enableCustomHttpDNS(true);
// Step 2: Set the HttpDNS resolution callback
TXLiveBase.setListener(new TXLiveBaseListener() {
    QOverride
    public void onCustomHttpDNS(String hostName, List<String> ipList) {
        // After resolving the hostName to an IP address, save it to the iPList and
        // MSDKDnsResolver is the HTTPDNS SDK resolution interface provided by Tenc
        String ips = MSDKDnsResolver.getInstance().getAddrByName(hostname);
        String[] ipArr = ips.split(";");
        if (0 != ipArr.length) {
            for (String ip : ipArr) {
                if ("0".equals(ip)) {
                    continue;
                }
                ipList.add(ip);
            }
        }
    }
});
```

## 7. HEVC Adaptive Downgrade Play

The player supports playing links that contain both HEVC and other video encoding formats, such as H.264. When the player device does not support the HEVC format, it will automatically downgrade to playing videos in the configured alternative encoding format (such as H.264).

Note: Supported from player version 11.7 of the premium version.





```
// Set backup playback link
String backupPlayUrl = "${backupPlayUrl}"; // Alternative playback link
mVodPlayer.setStringOption(TXVodConstants.VOD_KEY_MIMETYPE, TXVodConstants.VOD_PLAY
mVodPlayer.setStringOption(TXVodConstants.VOD_KEY_BACKUP_URL, backupPlayUrl); // Se
```

```
// Set the original HEVC playback link
String hevcPlayUrl = "${hevcPlayUrl}";
mVodPlayer.startVodPlay(hevcPlayUrl);
```

### 8. Volume Normalization

The player supports automatically adjusting the volume when playing audio to ensure that the volume of all audio is consistent. This can avoid problems with some audio being too loud or too quiet, providing a better auditory experience. Use TXVodPlayer#setAudioNormalization to set the volume normalization, with a loudness range of -70 to 0 (LUFS), and custom values are also supported. Note: Supported from player version 11.7 of the premium version.



/\*\*

Can be set to preset values (related classes or files: Android: TXVodConstants; iO Off: AUDIO\_NORMALIZATION\_OFF

On: AUDIO\_NORMALIZATION\_STANDARD (standard)
```
AUDIO_NORMALIZATION_LOW (low)

AUDIO_NORMALIZATION_HIGH (high)

Custom values can be set: from low to high, range -70 to 0 LUFS

*/

mVodPlayer.setAudioNormalization(TXVodConstants.AUDIO_NORMALIZATION_STANDARD); //

mVodPlayer.setAudioNormalization(TXVodConstants.AUDIO_NORMALIZATION_OFF); //
```

# Player Event Listening

You can bind a TXVodPlayListener listener to the TXVodPlayer object to use onPlayEvent (event notification) and onNetStatus (status feedback) to sync information to your application.

### Playback event notifications (onPlayEvent)

| Event ID                                  | Code | Description  |
|---|------|--|
| PLAY_EVT_PLAY_BEGIN                       | 2004 | Video playback started.  |
| PLAY_EVT_PLAY_PROGRESS                    | 2005 | The video playback progress (including<br>the current playback progress, loading<br>progress, and total video duration). |
| PLAY_EVT_PLAY_LOADING                     | 2007 | The video is being loaded. The<br>LOADING_END event will be reported if<br>video playback resumes.                       |
| PLAY_EVT_VOD_LOADING_END                  | 2014 | Video loading ended, and video playback resumed.   |
| TXVodConstants.VOD_PLAY_EVT_SEEK_COMPLETE | 2019 | Seeking was completed. The seeking feature is supported by v10.3 or later.   |
| VOD_PLAY_EVT_LOOP_ONCE_COMPLETE           | 6001 | A round of loop was completed. The loop feature is supported by v10.8 or later.  |
| TXVodConstants.VOD_PLAY_EVT_HIT_CACHE     | 2002 | Cache hit event during playback (supported by v11.2 or later).   |
| TXVodConstants.VOD_PLAY_EVT_VIDEO_SEI     | 2030 | Received SEI frame event (supported from player version 11.6 of the premium version).                                    |
| VOD_PLAY_EVT_HEVC_DOWNGRADE_PLAYBACK      | 2031 | HEVC downgrade playback occurs (supported by the player's advanced   |



|                                 |      | version 12.0).   |
|---------------------------------|------|--|
| VOD_PLAY_EVT_FIRST_VIDEO_PACKET | 2017 | The player receives the first frame data packet event (supported by version 12.0). |

### SEI frame

SEI (Supplemental Enhancement Information) frames are a type of frame used to transmit additional information. The premium version of the player will parse the SEI frames in the video stream and provide callbacks through the `VOD\_PLAY\_EVT\_VIDEO\_SEI` event. Note: Supported from player version 11.6 of the premium version.



| @Overric | de  |
|----------|---|
| public v | void onPlayEvent(TXVodPlayer player, int event, Bundle param) {                       |
| if       | (event == TXVodConstants.VOD_PLAY_EVT_VIDEO_SEI) {                                    |
|          | <pre>int seiType = param.getInt(TXVodConstants.EVT_KEY_SEI_TYPE); // the type</pre>   |
|          | <pre>int seiSize = param.getInt(TXVodConstants.EVT_KEY_SEI_SIZE); // the data s</pre> |
|          | <pre>byte[] seiData = param.getByteArray(TXVodConstants.EVT_KEY_SEI_DATA); // t</pre> |
| }        |   |
| }        |   |
|          |   |

#### Stop events

| Event ID                | Code  | Description  |
|-------------------------|-------|--|
| PLAY_EVT_PLAY_END       | 2006  | Video playback ended.  |
| PLAY_ERR_NET_DISCONNECT | -2301 | The network was disconnected and could not be reconnected after multiple retries. You can restart the player to perform more connection retries. |
| PLAY_ERR_HLS_KEY        | -2305 | Failed to get the HLS decryption key.  |

### Warning events

You can ignore the following events, which are only used to notify you of some internal events of the SDK.

| Event ID                          | Code | Description  |
|-----------------------------------|------|--|
| PLAY_WARNING_VIDEO_DECODE_FAIL    | 2101 | Failed to decode the current video frame.  |
| PLAY_WARNING_AUDIO_DECODE_FAIL    | 2102 | Failed to decode the current audio frame.  |
| PLAY_WARNING_RECONNECT            | 2103 | The network was disconnected, and automatic reconnection was performed (the PLAY_ERR_NET_DISCONNECT event will be thrown after three failed attempts). |
| PLAY_WARNING_HW_ACCELERATION_FAIL | 2106 | Failed to start the hardware decoder, and the software decoder was used instead.   |

### **Connection events**

The following server connection events are mainly used to measure and collect the server connection time:

| Event ID                   | Code | Description   |  |  |
|----------------------------|------|---|--|--|
| PLAY_EVT_VOD_PLAY_PREPARED | 2013 | The player has been prepared and can start playback. If |  |  |
|                            |      | autoPlay is set to false , you need to call             |  |  |

|                            |      | resume after receiving this event to start playback.               |
|----------------------------|------|--|
| PLAY_EVT_RCV_FIRST_I_FRAME | 2003 | The network received the first renderable video data packet (IDR). |

### Image quality events

The following events are used to get image change information:

| Event ID                   | Code | Description                   |
|----------------------------|------|-------------------------------|
| PLAY_EVT_CHANGE_RESOLUTION | 2009 | The video resolution changed. |
| PLAY_EVT_CHANGE_ROTATION   | 2011 | The MP4 video was rotated.    |

#### Video information events

| Event ID                                   | Code | Description  |
|--|------|--|
| TXLiveConstants.PLAY_EVT_GET_PLAYINFO_SUCC | 2010 | Obtained the information of the file played back successfully. |

If you play back a video through fileId  $\$  and the playback request succeeds (called

API:: startVodPlay(TXPlayInfoParams playInfoParams) ), the SDK will notify the upper layer of some

request information, and you can parse param to get the video information after receiving the

TXLiveConstants.PLAY\_EVT\_GET\_PLAYINFO\_SUCC event.

| Video Information                           | Description  |
|---|--|
| EVT_PLAY_COVER_URL                          | Video thumbnail URL  |
| EVT_PLAY_URL                                | Video playback address   |
| EVT_PLAY_DURATION                           | Video duration   |
| EVT_DESCRIPTION                             | Event description  |
| EVT_PLAY_NAME                               | Video name   |
| TXVodConstants.EVT_IMAGESPRIT_WEBVTTURL     | Download URL of the image sprite WebVTT file, which is supported by v10.2 or later |
| TXVodConstants.EVT_IMAGESPRIT_IMAGEURL_LIST | The download URL of the image sprite image, which is supported by v10.2 or later.  |
| TXVodConstants.EVT_DRM_TYPE                 | The encryption type, which is supported by v10.2 or later.                         |



TXVodConstants.EVT\_KEY\_WATER\_MARK\_TEXT

Ghost watermark text content (supported from version 11.6).

Below is the sample code of using <code>onPlayEvent</code> to get the video playback information:



```
mVodPlayer.setVodListener(new ITXVodPlayListener() {
    @Override
    public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
        if (event == TXLiveConstants.PLAY_EVT_VOD_PLAY_PREPARED) {
            // The player preparation completion event is received, and you can cal
            // The player preparation completion event is received, and you can cal
            // The player preparation completion event is received, and you can cal
            // The player preparation completion event is received, and you can cal
            // The player preparation completion event is received, and you can cal
            // The player preparation completion event is received, and you can cal
            // The player preparation completion event is received, and you can cal
            // The player player player player player player, interval
            // The player preparation completion event is received, and you can cal
            // The player player, interval
            // The player playe
```

### Ghost watermark

The content of the ghost watermark is filled in the player signature and is ultimately displayed on the playback end through collaboration between the cloud and the player, ensuring the security of the watermark throughout the transmission process. Follow the tutorial to configure the ghost watermark in the player signature. The content of the ghost watermark can be obtained through `param.getString(TXVodConstants.EVT\_KEY\_WATER\_MARK\_TEXT)` after receiving the `TXVodConstants#VOD\_PLAY\_EVT\_GET\_PLAYINFO\_SUCC` event from the player. For detailed usage tutorial, please refer to SuperPlayer Component > Ghost Watermark. Note: Supported from player version 11.6.

#### **Playback error event**

#### Note:

Error events [-6004, -6010] are supported starting from version 11.0.

| Event ID                          | Value | Description  |
|-----------------------------------|-------|--|
| PLAY_ERR_NET_DISCONNECT           | -2301 | Video data error that cannot be recovered by<br>retrying. For example, network anomaly or<br>data download error that causes<br>demultiplexing timeout or failure. |
| PLAY_ERR_HLS_KEY                  | -2305 | Failed to obtain HLS decryption key.   |
| VOD_PLAY_ERR_SYSTEM_PLAY_FAIL     | -6004 | System player playback error.  |
| VOD_PLAY_ERR_DECODE_VIDEO_FAIL    | -6006 | Video decoding error, video format not supported.  |
| VOD_PLAY_ERR_DECODE_AUDIO_FAIL    | -6007 | Audio decoding error, audio format not supported.  |
| VOD_PLAY_ERR_DECODE_SUBTITLE_FAIL | -6008 | Subtitle decoding error.   |
| VOD_PLAY_ERR_RENDER_FAIL          | -6009 | Video rendering error.   |
|                                   |       |  |

| VOD_PLAY_ERR_PROCESS_VIDEO_FAIL | -6010 | Video post-processing error.  |
|---------------------------------|-------|---|
| VOD_PLAY_ERR_GET_PLAYINFO_FAIL  | -2306 | Failed to obtain the on-demand file<br>information. It is recommended to check<br>whether the AppId, FileId or Psign is filled in<br>correctly. |

### Playback status feedback (onNetStatus)

The status feedback is triggered once every 0.5 seconds to provide real-time feedback on the current status of the pusher. It can act as a dashboard to inform you of what is happening inside the SDK so you can better understand the current video playback status.

| Parameter                | Description   |
|--------------------------|---|
| NET_STATUS_CPU_USAGE     | Current instantaneous CPU utilization   |
| NET_STATUS_VIDEO_WIDTH   | Video resolution - width  |
| NET_STATUS_VIDEO_HEIGHT  | Video resolution - height   |
| NET_STATUS_NET_SPEED     | Current network data reception speed in KBps  |
| NET_STATUS_VIDEO_FPS     | Current video frame rate of streaming media   |
| NET_STATUS_VIDEO_BITRATE | Current video bitrate in bps of streaming media                                       |
| NET_STATUS_AUDIO_BITRATE | Current audio bitrate in bps of streaming media                                       |
| NET_STATUS_VIDEO_CACHE   | Buffer (`jitterbuffer`) size. If the current buffer length is 0, lag will occur soon. |
| NET_STATUS_SERVER_IP     | Connected server IP   |

Below is the sample code of using `onNetStatus` to get the video playback information:





```
mVodPlayer.setVodListener(new ITXVodPlayListener() {
    @Override
    public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
    }
    @Override
    public void onNetStatus(TXVodPlayer player, Bundle bundle) {
        // Get the current CPU utilization
        CharSequence cpuUsage = bundle.getCharSequence(TXLiveConstants.NET_STATUS_C
        // Get the video width
        int videoWidth = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_WIDTH);
```

```
// Get the video height
        int videoHeight = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_HEIGHT);
        // Get the real-time rate in Kbps
        int speed = bundle.getInt(TXLiveConstants.NET_STATUS_NET_SPEED);
        // Get the current video frame rate of streaming media
        int fps = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_FPS);
        // Get the current video bitrate in bps of streaming media
        int videoBitRate = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_BITRATE);
        // Get the current audio bitrate in bps of streaming media
        int audioBitRate = bundle.getInt(TXLiveConstants.NET STATUS AUDIO BITRATE);
        // Get the buffer (`jitterbuffer`) size. If the current buffer length is 0,
        int jitterbuffer = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_CACHE);
        // Get the connected server IP
        String ip = bundle.getString(TXLiveConstants.NET_STATUS_SERVER_IP);
    }
});
```

# Other functions

### HLS live video source playback

The premium version of the player supports playing HLS live video sources. Starting from version 11.8, it supports live video sources with HLS EVENT. Usage is as follows:





```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setMediaType(TXVodConstants.MEDIA_TYPE_HLS_LIVE); // Specify the HLS live me
mVodPlayer.setConfig(config);
mVodPlayer.startVodPlay(${YOUR_HSL_LIVE_URL});
```

### Scenario-Specific Features

### 1. SDK-based demo component

Based on the Player SDK, Tencent Cloud has developed a player component. It integrates quality monitoring, video encryption, Top Speed Codec, definition selection, and small window playback and is suitable for all VOD and live playback scenarios. It encapsulates complete features and provides upper-layer UIs to help you quickly create a playback program comparable to popular video apps.

### 2. Open-source GitHub projects

Based on the Player SDK, Tencent Cloud has developed immersive video player, video feed stream, and multi-layer reuse components and will provide more user scenario-based components on future versions. You can download Player for Android to try them out.

# Flutter Integration Integration Guide

Last updated : 2024-04-26 11:09:31

# **Environment Requirements**

Flutter 3.0 or later Developing for Android: Android Studio 3.5 or later. Devices with Android 4.1 or later. Developing for iOS: Xcode 11.0 or later. OS X 10.11 or later. A valid developer signature for your project.

### SDK Download

You can download the Player SDK for Flutter here.

### Note:

To run this demo, you need to set your own player license in the demo\_config, and modify the package name and bundleld to your signed package name and bundleld in the Android and iOS configurations.

### **Quick Integration**

### Adding the following dependencies to the pubspec.yaml of your project

You can integrate LiteAVSDK\_Player or LiteAVSDK\_Professional as needed.

1. To integrate the latest version of LiteAVSDK\_Player\_Premium (player premium version), add configuration in pubspec.yaml :



```
super_player:
git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: Player_Premium
```

Integrate the latest version of LiteAVSDK\_Player (which is integrated by default) and add the following configuration to pubspec.yaml :





```
super_player:
git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
```

To integrate the latest version of LiteAVSDK\_Professional , change the configuration in pubspec.yaml as follows:



```
super_player:
git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: Professional
```

To integrate a specified version of the player SDK, specify the corresponding version through the tag that ref depends on as follows:



```
super_player:
git:
   url: https://github.com/LiteAVSDK/Player_Flutter
   path: Flutter
   ref: release_player_v1.0.6
```

# `release\_player\_v1.0.6` indicates to integrate TXLiteAVSDK\_Player\_10.6.0.11182 fo

For more archived tags, see Release List.

2. After the integration, you can obtain the Flutter dependencies through the UI that comes with the code editor or by directly running the following command:



flutter pub get

3. During use, you can run the following command to update existing Flutter dependencies:



flutter pub upgrade

### Adding native configuration

### Android configuration

1. Add the following configuration to AndroidManifest.xml .





```
<!--network permission-->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<!--storage-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
</uses-permission andr
```

Network security configuration allows the app to send HTTP requests

### 🕗 Tencent Cloud

For security reasons, starting from Android P, Google requires that all requests made by the app use encrypted connections. The player SDK will start a local server to proxy HTTP requests. **If your app's targetSdkVersion is greater than or equal to 28**, you can use network security configuration to allow HTTP requests to be sent to 127.0.0.1. Otherwise, you will encounter the error "java.io.IOException: Cleartext HTTP traffic to 127.0.0.1 not permitted" when playing videos, which will prevent the video from playing. The configuration steps are as follows: 1.1 Create a new file network\_security\_config.xml under res/xml in your project and configure the network security settings.



<?xml version="1.0" encoding="utf-8"?> <network-security-config>



1.2 Add the following attributes to the application tag in the AndroidManifest.xml file:



```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
<application android:networkSecurityConfig="@xml/network_security_config"
... >
```



```
</application> </manifest>
```

2. Make sure that the build.gradle in the Android directory uses mavenCenter and can successfully download dependencies.



repositories { mavenCentral() }

3. Configure the minimum SDK version for Android. Due to the default configuration of Flutter, the minimum version of Android is too low. You need to manually change it to at least 19. If you want to use the picture-in-picture feature, the compileSdkVersion and targetSdkVersion need to be changed to at least 31.



```
compileSdkVersion 31
defaultConfig {
    applicationId "com.tencent.liteav.demo"
    minSdkVersion 19
    targetSdkVersion 31
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
```



}

4. Add the following configuration xmlns:tools="http://schemas.android.com/tools" to the root manifest tag in the AndroidManifest.xml file. Here is an example:



```
<manifest

<manifest

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

package="com.example.player">

<!-- your config..... -->

</manifest>
```



Add tools:replace="android:label" to the application node. Here is an example:



```
<application
android:label="super_player_example"
android:icon="@mipmap/ic_launcher"
android:requestLegacyExternalStorage="true"
tools:replace="android:label">
<!-- your config..... -->
</application>
```

5. To update the dependency version of the native SDK, manually delete the build folder in the Android directory or run the following command to force a refresh.



./gradlew build

#### iOS configuration

Note: iOS currently does not support project running and debugging in simulators; therefore, we recommend you develop and debug your project on a real device.

1. Add the following configuration to the Info.plist file of iOS:



```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key>
<true/>
</dict>
```

2. iOS natively uses pod for dependency. Edit the podfile file and specify your player SDK edition. TXLiteAVSDK\_Player\_Premium is integrated as follows.



pod 'TXLiteAVSDK\_Player\_Premium'
#pod 'TXLiteAVSDK\_Player'

//Player\_Premium Edition
// Player Edition

Integrate LiteAVSDK\_Professional :





```
pod 'TXLiteAVSDK_Professional' // Professional Edition
```

If you do not specify the edition, the latest version of TXLiteAVSDK\_Player will be installed.

3. In some cases such as new version release, you need to forcibly update the iOS player dependencies by running the following command in the iOS directory:



rm -rf Pods rm -rf Podfile.lock pod update

#### 4. Landscape configuration

If your app needs to support landscape mode, you need to set the supported orientations for portrait and landscape in the Deployment Info tab on the General page of the IOS project configuration. You can check all options as shown in the figure below:



|                   | General | Signing & Capabilities | Resource Tags      | Info                         | Build Settings   | Build Phases | Build Rule |
|-------------------|---------|------------------------|--------------------|------------------------------|--|--------------|------------|
| ✓ Deployment Info |         |                        |                    |                              |  |              |            |
|                   |         |                        | iPhone Orientation | ✓ Po<br>□ Up<br>✓ La<br>✓ La | ortrait<br>oside Down<br>ndscape Left<br>ndscape Right |              |            |
|                   |         |                        | iPad Orientation   | ✓ Pc<br>✓ Up<br>✓ La         | ortrait<br>oside Down<br>ndscape Left<br>ndscape Right |              |            |

#### 5. Picture-in-Picture configuration

If your project needs to support picture-in-picture, you need to check the "Audio, AirPlay, and Picture in Picture" option under the Background Modes tab on the Signing & Capabilities page of the IOS project configuration to enable the picture-in-picture capability for your project, as shown in the figure below:

|   |            |        |         |         | General | Signing & Capabilities | Resour | ce Tags  | Info   | Build Setting:   |
|---|------------|--------|---------|---------|---------|------------------------|--------|--|--|--|
| + | Capability | All    | Debug   | Release | Profile |                        |        |  |  |  |
|   | > Signing  |        |         |         |         |                        |        |  |  |  |
|   | 🗸 😂 Back   | ground | d Modes |         |         |                        |        |  |  |  |
|   |            |        |         |         |         |                        | Modes  | <ul> <li>Audio,</li> <li>Locati</li> <li>Voice</li> <li>Extern</li> <li>Uses I</li> <li>Acts a</li> <li>Backg</li> <li>Remot</li> <li>Backg</li> <li>Uses I</li> <li>Push t</li> </ul> | AirPlay,<br>on upda<br>over IP<br>al acces<br>Bluetoot<br>s a Blue<br>round fe<br>te notific<br>round pr<br>Nearby In<br>to Talk | and Picture in I<br>tes<br>sory communic<br>h LE accessorie<br>tooth LE access<br>tch<br>ations<br>rocessing<br>hteraction |

### Integrating the Video Playback License

If you have obtained a license, you can view the license URL and key in the RT-Cube console.

If you don't have the required license yet, you can get it as instructed in Adding and Renewing a License.

Before you integrate the player, you need to sign up for a Tencent Cloud account, apply for the video playback license,

and configure the license as follows (we recommend you do this when the application is launched):

If you don't configure a license, errors may occur during playback.



```
String licenceURL = ""; // The license URL obtained
String licenceKey = ""; // The license key obtained
SuperPlayerPlugin.setGlobalLicense(licenceURL, licenceKey);
```

### **Custom Development**



The Flutter plugin of the Player SDK is based on the system's native player capabilities. You can use either of the following methods for custom development:

Use the VOD playback API class TXVodPlayerController or live playback API class

TXLivePlayerController for custom development. You can refer to the demos we provide

(  ${\tt DemoTXVodPlayer}$  and  ${\tt DemoTXLivePlayer}$  in example ).

The player component SuperPlayerController encapsulates VOD playback and live playback capabilities and includes basic UI elements.

You can copy the code of the player component (in example/lib/superplayer ) to your project for custom development.

## FAQs

1. Errors about missing APIs such as No visible @interface for 'TXLivePlayer' declares the selector 'startLivePlay:type:' occur on iOS.
Run the following command to update the iOS SDK:





rm -rf Pods rm -rf Podfile.lock pod update

2. SDK or symbol conflicts occur when tencent\_trtc\_cloud and Flutter player are integrated at the same time. Common exception log: java. lang.RuntimeException: Duplicate class com.tencent.liteav.TXLiteAVCode found in modules classes.jar

In this case, you need to integrate the Professional version of the Flutter player, so that tencent\_trtc\_cloud and the Flutter player both depend on the same version of LiteAVSDK\_Professional .



For example, to depend on TXLiteAVSDK\_Professional\_10.3.0.11196 for Android or

TXLiteAVSDK\_Professional to 10.3.12231 for iOS, the dependency declaration is as follows:



```
tencent_trtc_cloud: 2.3.8
super_player:
  git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: release_pro_v1.0.3.11196_12231
```

### 🔗 Tencent Cloud

3. When multiple player instances need to be used at the same time, the video image gets blurry when videos are frequently switched.

When each player component container is terminated, call the dispose method of the player to release the player. 4. Other common issues with Flutter dependencies:

Run the flutter doctor command to check the runtime environment until "No issues found!" is displayed.

Run flutter pub get to ensure that all dependent components have been updated successfully.

5. After integrating SuperPlayer, you may encounter the following manifest error:



Attribute application@label value=(super\_player\_example) from AndroidManifest.xml:9 is also present at [com.tencent.liteav:LiteAVSDK\_Player:10.8.0.13065] AndroidManife


Suggestion: add 'tools:replace="android:label"' to <application> element at Android

**Solution:** The SuperPlayer Android SDK has already defined the label attribute in its AndroidManifest, and when you create a new Flutter project, the AndroidManifest in the Android directory also defines the label attribute. Therefore, it is recommended that you follow the error message and

add xmlns:tools="http://schemas.android.com/tools" to the root manifest tag in your Android project directory's AndroidManifest.xml. Then, add tools:replace="android:label" to

the application node. This will override the label attribute and resolve the conflict.

6. After integrating SuperPlayer, you may encounter version conflicts with other dependencies. The error message may look like this:





uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in library [:s

**Solution**: Currently, the minimum supported version for the SuperPlayer Android SDK is Android 19, while some versions of Flutter default to Android 16 as the minimum supported version. It is recommended that you raise the minimum supported version to Android 19. To do this, go to the main module of your Android project, which is usually the app directory, and modify the minSdkVersion in the build.gradle file to 19.

7. How to extract the runtime logs of the player SDK

**Solution**: The player SDK outputs the runtime logs to a local file by default. Tencent Cloud technical support may need these logs to diagnose issues. On the Android platform, the logs are saved in the

directory /sdcard/Android/data/packagename/files/log/tencent/liteav , while on the iOS

 $platform, the logs are saved in the \verb| sandbox/Documents/log| directory.$ 

8. How to reduce console log output

Solution: You can set the log output level using the following

interface: SuperPlayerPlugin.setLogLevel(TXLogLevel.LOG\_LEVEL\_NULL). The following log levels are supported:





```
class TXLogLevel {
  static const LOG_LEVEL_VERBOSE = 0; // Output logs of all levels.
  static const LOG_LEVEL_DEBUG = 1; // Output logs of DEBUG, INFO, WARNING, ERROR,
  static const LOG_LEVEL_INFO = 2; // Output logs of INFO, WARNING, ERROR, and FATA
  static const LOG_LEVEL_WARN = 3; // Output logs of WARNING, ERROR, and FATAL leve
  static const LOG_LEVEL_ERROR = 4; // Output logs of ERROR and FATAL levels.
  static const LOG_LEVEL_FATAL = 5; // Only output logs of FATAL level.
  static const LOG_LEVEL_NULL = 6; // Do not output any SDK logs.
}
```

9. During the use of the project, native-related errors may occur, such as:

## 🕗 Tencent Cloud

Errors :'incompatible types', error: initializing 'BOOL' (aka 'bool') with an expression of incompatible type 'void', etc., are caused by SDK updates, which make the SDK incompatible with the native code in the Flutter end. In this case, you only need to update the SDK version. **Solution**: In the project directory, open the terminal and enter the following commands in sequence:



flutter pub cache clean flutter clean flutter pub upgrade flutter pub get



Make sure the commands execute successfully to update the local Flutter dependencies.

Then, in the ios directory, open the terminal and enter the following command to update the iOS dependencies:



```
rm -rf Pods
rm -rf Podfile.lock
pod update
```

If the problem still persists, you can try deleting the project's build folder and manually deleting the Flutter dependency cache folder .pub-cache on your computer. Then refresh the Flutter pub dependencies and compile and run the project again.

10. When playing a video on the Android on-demand player, the edges of the player appear tiled and stretched.

This issue is caused by the texture rendering problem of the flutter end sdk. You can upgrade the flutter version to flutter 3.7.0 or above.

11. The flutter debugging and test package runs without any problems, but the official package crashes when it is installed.

Flutter uses obfuscation by default when building a formal package. The player SDK needs to configure the following obfuscation rules:



-keep class com.tencent.\*\* { \*; }

## 🔗 Tencent Cloud

12. Unable to play local videos

The flutter player supports local video playback. You need to pass the correct local video address to the video playback interface. If you encounter playback issues, first check if the local video address is available and if the file is damaged. If the local video is fine, check if the app has storage or image/video reading permissions.

13. When running the iOS project, errors such as CocoaPods could not find compatible versions for pod "Flutter" or similar are reported.

This issue occurs because the high flutter development environment no longer supports lower iOS versions. You can check if the iOS version configured for Minimum Deployments in the project is too low or if it inherits dependencies that only support lower iOS versions.

## More

You can try out all the features by running the example in the project.

The player SDK website provides demos for iOS, Android, and web. Click here to use them.

# **VOD** Scenario

Last updated : 2024-04-26 11:09:31

## Intended Audience

This document describes some of the proprietary capabilities of Tencent Cloud. Make sure that you have activated the relevant Tencent Cloud services before reading this document. If you don't have an account yet, sign up for free trial first.

## This Document Describes

How to integrate the Tencent Cloud RT-Cube Player SDK for Flutter.

How to use the Player SDK for VOD playback.

How to use the underlying capabilities of the Player SDK to implement more features.

## Basics

This document describes the VOD playback feature of the Player SDK. You can start by understanding the following basics:

### Live streaming and video on demand

In live streaming, the video source is pushed by the host in real time. When the host stops pushing the stream, the player will also stop playing the video. Because the live stream is played back in real time, no progress bar will be displayed in the player during the playback.

In video on demand (VOD), the video source is a video file in the cloud, which can be played back at any time as long as it is not deleted from the cloud. A progress bar is displayed for controlling the playback progress. Typical VOD scenarios including viewing videos on video websites such as Tencent Video and Youku Tudou.

### Supported protocols

Common VOD protocols are as listed below. Currently, VOD URLs in HLS format (starting with http and ending with .m3u8) are popular.

## Notes

The Player SDK **does not impose any limits on the sources of playback URLs**, which means that you can use it to play back videos from both Tencent Cloud and non-Tencent Cloud URLs. However, players in the SDK support



only live streaming URLs in FLV, RTMP, and HLS (M3U8) formats, as well as VOD URLs in MP4, HLS (M3U8), and FLV formats.

## **SDK** Integration

## Step 1. Integrate the SDK ZIP file

Download and integrate the SDK ZIP file as instructed in Integration Guide.

### Step 2. Create a controller



TXVodPlayerController \_controller = TXVodPlayerController();

## Step 3. Configure event listening





```
// Listen for the video width and height change and set an appropriate aspect ratio
_controller.onPlayerNetStatusBroadcast.listen((event) async {
    double w = (event["VIDEO_WIDTH"]).toDouble();
    double h = (event["VIDEO_HEIGHT"]).toDouble();
    if (w > 0 && h > 0) {
      setState(() {
        _aspectRatio = 1.0 * w / h;
      });
    };
}
```

## Step 4. Add a layout



```
@override
Widget build(BuildContext context) {
return Container(
   decoration: BoxDecoration(
        image: DecorationImage(
            image: AssetImage("images/ic_new_vod_bg.png"),
            fit: BoxFit.cover,
```

```
)),
 child: Scaffold(
     backgroundColor: Colors.transparent,
      appBar: AppBar(
       backgroundColor: Colors.transparent,
        title: const Text('VOD'),
      ),
     body: SafeArea(
         child: Container(
            height: 150,
            color: Colors.black,
            child: Center(
              child: _aspectRatio > 0
                  ? AspectRatio(
                aspectRatio: _aspectRatio,
                child: TXPlayerVideo(controller: _controller),
              ) : Container(),
            ),
          ))));
}
```

## Step 5. Initialize the player





// Initialize the player and assign the shared texture
await \_controller.initialize();

## Step 6. Start the playback

Through the URL

Through fileId

TXVodPlayerControllerwill internally recognize the playback protocol automatically. You only need to pass inyour playback URL to the startVodPlayfunction.





```
// Play back the video resource
String _url =
    "http://1400329073.vod2.myqcloud.com/d62d88a7vodtranscq1400329073/59c68fe752858
await _controller.startVodPlay(_url);
```





Find the target video file in Media Assets, and you can view the FileId below the filename. Play back the video through the FileId , and the player will request the backend for the real playback URL. If the network is abnormal or the FileId doesn't exist, the TXLiveConstants.PLAY\_ERR\_GET\_PLAYINFO\_FAIL event will be received; otherwise, TXLiveConstants.PLAY\_EVT\_GET\_PLAYINFO\_SUCC will be received, indicating that the request succeeded.

## Step 7. Stop the playback

**Remember to call the controller termination method** when stopping the playback, especially before the next call of startVodPlay . This can prevent memory leak and screen flashing issues.



```
@override
void dispose() {
   _controller.dispose();
```

```
super.dispose();
}
```

# **Basic Feature Usage**

1. Playback control

Starting playback





```
Video on Demand
```

```
// Start playback
_controller.startVodPlay(url)
```

#### Pausing playback



// Pause the video
\_controller.pause();

**Resuming playback** 



// Resume the video
\_controller.resume();

## Stopping playback



// Stop the video
\_controller.stopPlay(true);

Stopping the player





// Release the controller
\_controller.dispose();

## Adjusting playback progress (seek)

When the user drags the progress bar, seek can be called to start playback at the specified position. The Player SDK supports accurate seek.





double time = 600; // The value is of `double` type and is in seconds. // Adjust the playback progress \_controller.seek(time);

#### Seek to the specified Program Date Time (PDT) point in the video stream

To seek to the specified Program Date Time (PDT) point in the video stream, which enables functions such as fastforward, rewind, and progress bar jumping, currently only HLS video format is supported. Note: Starting from version 11.6 of the player's premium edition, this function is supported.



int pdtTimeMs = 600; // Unit is milliseconds
\_controller.seekToPdtTime(time);

### Specifying playback start time

You can specify the playback start time before calling startVodPlay for the first time.





```
double startTimeInSecond = 60; // Unit: Second.
_controller.setStartTime(startTimeInSecond); // Set the playback start time
_controller.startVodPlay(url);
```

### 2. Adjustable-Speed playback

The VOD player supports adjustable-speed playback. You can use the setRate API to set the VOD playback speed, such as 0.5x, 1.0x, 1.2x, and 2x speed.



// Set playback at 1.2X rate
\_controller.setRate(1.2);

## 3. Playback loop



// Set playback loop \_controller.setLoop(true); // Get the current playback loop status \_controller.isLoop();

## 4. Muting/Unmuting





// Mute or unmute the player. true: Mute; false: Unmute
\_controller.setMute(true);

## 5. Roll image ad

The Player SDK allows you to add roll images on the UI for advertising as follows:

If autoPlay is set to NO, the player will load the video normally but will not immediately start playing it back. Users can see the roll image ad on the player UI after the player is loaded and before the video playback starts. When the ad display stop conditions are met, the resume API will be called to start video playback.





```
_controller.setAutoPlay(false); // Set manual playback
_controller.startVodPlay(url); // The video will be loaded after `startVodPlay`
// .....
// Display the ad on the player UI
// .....
_controller.resume(); // Call `resume` to start playing back the video after the a
```

### 6. HTTP-REF

## 🕗 Tencent Cloud

headers in TXVodPlayConfig can be used to set HTTP request headers, such as the Referer field commonly used to prevent the URL from being copied arbitrarily (Tencent Cloud provides a more secure signature-based hotlink protection solution) and the Cookie field for client authentication.



```
FTXVodPlayConfig playConfig = FTXVodPlayConfig();
Map<String, String> httpHeaders = {'Referer': 'Referer Content'};
playConfig.headers = httpHeaders;
_controller.setConfig(playConfig);
```

### 7. Hardware acceleration

It is extremely difficult to play back videos of the Blu-ray (1080p) or higher image quality smoothly if only software decoding is used. Therefore, if your main scenario is game live streaming, we recommend you use hardware acceleration.

Before switching between software and hardware decoding, you need to call **stopPlay** first. After the switch, you need to call **startVodPlay**; otherwise, severe blurs will occur.



```
_controller.stopPlay(true);
_controller.enableHardwareDecode(true);
_controller.startVodPlay(url);
```



## 8. Definition settings

The SDK supports the multi-bitrate format of HLS, so users can switch between streams at different bitrates to switch the video definition. You can set the definition as follows:



List \_supportedBitrates = (await \_controller.getSupportedBitrates())!;; // Get the int index = \_supportedBitrates[i]; // Specify the subscript of the bitrate of the \_controller.setBitrateIndex(index); // Switch to the stream at the target bitrate

During playback, you can call \_\_controller.setBitrateIndex(int) at any time to switch the bitrate. During switch, the data of another stream will be pulled. The SDK is optimized for Tencent Cloud multi-bitrate files to



implement smooth switch.

If you know the resolution information of the video stream in advance, you can specify the video resolution to be played before starting playback to avoid switching streams after playback. For detailed methods, refer to Player Configuration#Specifying Resolution Before Playback.

## 9. Adaptive bitrate streaming

The SDK supports adaptive bitrate streaming of HLS. After this capability is enabled, the player can dynamically select the most appropriate bitrate for playback based on the current bandwidth. You can enable adaptive bitrate streaming as follows:



```
_controller.setBitrateIndex(-1); // Pass in `-1` for the `index` parameter
```

During playback, you can call \_controller.setBitrateIndex(int) at any time to switch to another bitrate. After the switch, adaptive bitrate streaming will be disabled.

## 10. Enabling smooth bitrate switching

Before starting playback, you can enable smooth bitrate switching to seamlessly switch between different definitions (bitrates) during playback. If smooth bitrate switch is enabled, the transition between different bitrates will be smoother but will be more time-consuming. Therefore, this feature can be configured as needed.



```
FTXVodPlayConfig playConfig = FTXVodPlayConfig();
/// If it is set to `true`, the bitrate can be switched smoothly. If it is set to `
playConfig.smoothSwitchBitrate = true;
_controller.setConfig(playConfig);
```

## 11. Playback progress listening

There are two metrics for the VOD progress: **loading progress** and **playback progress**. Currently, the SDK notifies the two progress metrics in real time through event notifications.

You can use the onPlayerEventBroadcast API to listen on player events, and the progress notification will be called back to your application through the **PLAY\_EVT\_PLAY\_PROGRESS** event.





```
_controller.onPlayerEventBroadcast.listen((event) async {
    if(event["event"] == TXVodPlayEvent.PLAY_EVT_PLAY_PROGRESS) {// For more informat
    // Playable duration, i.e., loading progress, in milliseconds
    double playableDuration = event[TXVodPlayEvent.EVT_PLAYABLE_DURATION_MS].toDoub
    // Playback progress in seconds
    int progress = event[TXVodPlayEvent.EVT_PLAY_PROGRESS].toInt();
    // Total video duration in seconds
    int duration = event[TXVodPlayEvent.EVT_PLAY_DURATION].toInt();
    }
});
```
# 12. Playback network speed listening

You can use the onPlayerNetStatusBroadcast API to listen on the player network status such as NET\_STATUS\_NET\_SPEED .



```
_controller.onPlayerNetStatusBroadcast.listen((event) {
    (event[TXVodNetEvent.NET_STATUS_NET_SPEED]).toDouble();
});
```

# 13. Video resolution acquisition

The Player SDK plays back a video through a URL string. The URL doesn't contain the video information, and you need to access the cloud server to load such information. Therefore, the SDK can only send the video information to your application as event notifications.

You can get the resolution information in the following two methods:

Method 1: Use theNET\_STATUS\_VIDEO\_WIDTHandNET\_STATUS\_VIDEO\_HEIGHTofonPlayerNetStatusBroadcastto get the video width and height.

Method 2: Directly call getWidth() and getHeight() to get the current video width and height after receiving the PLAY\_EVT\_VOD\_PLAY\_PREPARED event callback from the player.





```
_controller.onPlayerNetStatusBroadcast.listen((event) {
   double w = (event[TXVodNetEvent.NET_STATUS_VIDEO_WIDTH]).toDouble();
   double h = (event[TXVodNetEvent.NET_STATUS_VIDEO_HEIGHT]).toDouble();
});
// Get the video width and height. The values can be returned only after the `PLAY_
_controller.getWidth();
_controller.getHeight();
```

## 14. Player buffer size

During normal video playback, you can control the maximum size of the data buffered from the network in advance. If the maximum buffer size is not configured, the player will use the default buffer policy to guarantee a smooth playback experience.





```
FTXVodPlayConfig playConfig = FTXVodPlayConfig();
playConfig.maxBufferSize = 10; /// The maximum buffer size during playback in MB
_controller.setConfig(playConfig);
```

## 15. Local video cache

In short video playback scenarios, the local video file cache is required, so that general users don't need to consume traffic again to reload an already watched video.

Supported format: The SDK supports caching videos in two common VOD formats: HLS (M3U8) and MP4.

**Enablement time:** The SDK doesn't enable the caching feature by default. We recommend you do not enable it for scenarios in which most videos are watched only once.

**Enablement method:** This feature can be enabled in the player and takes effect globally. To enable it, you need to configure two parameters: local cache directory and cache size.



// Set the global cache directory and cache size in MB of the playback engine
SuperPlayerPlugin.setGlobalMaxCacheSize(200);
// Set the global cache directory of the playback engine
SuperPlayerPlugin.setGlobalCacheFolderPath("postfixPath");

# 16. External Subtitles

Note: This feature requires Player Premium version 11.7 to be supported.

The premium version of the player SDK supports adding and switching external subtitles, and now supports two subtitle formats: SRT and VTT.

Best practice: It is recommended to add subtitles and configure subtitle styles before calling startVodPlay. After receiving the VOD\_PLAY\_EVT\_VOD\_PLAY\_PREPARED event, call selectTrack to choose the subtitle. Adding subtitles does not automatically load them. After calling selectTrack, the subtitles will be loaded. The successful selection of subtitles will trigger the VOD\_PLAY\_EVT\_SELECT\_TRACK\_COMPLETE event callback.When a subtitle is selected, the subtitle text content is passed through TXVodPlayEvent.EVENT\_SUBTITLE\_DATA event callback, the display of subtitles needs to be handled by the business side.

### Step 1: Add external subtitles.





// Add external subtitles, pass in the subtitle url, subtitle name, subtitle type, controller.addSubtitleSource("https://mediacloud-76607.gzc.vod.tencent-cloud.com/De

```
// After starting to play the video, monitor the subtitle text content callback
_controller.onPlayerEventBroadcast.listen((event) async {
    if(event["event"] == TXVodPlayEvent.EVENT_SUBTITLE_DATA) {
        // Subtitle text content, can be used for display
        String subtitleDataStr = event[TXVodPlayEvent.EXTRA_SUBTITLE_DATA] ?? "";
    }
});
```



Step 2: Switch subtitles after playback starts.



```
// After starting to play the video, select the added external subtitles and call i
_controller.onPlayerEventBroadcast.listen((event) async {
    if(event["event"] == TXVodPlayEvent.PLAY_EVT_VOD_PLAY_PREPARED) {
      List<TXTrackInfo> trackInfoList = await _vodPlayerController.getSubtitleTrackI
      for (TXTrackInfo tempInfo in trackInfoList) {
         if(tempInfo.name == "subtitleName") {
            //Select subtitles
            _vodPlayerController.selectTrack(tempInfo.trackIndex);
```

```
} else {
    // If other subtitles are not needed, perform deselectTrack
    __vodPlayerController.deselectTrack(tempInfo.trackIndex);
    }
    }
};
// If necessary, you can listen for track switching messages
_controller.onPlayerEventBroadcast.listen((event) async {
    if(event["event"] == TXVodPlayEvent.VOD_PLAY_EVT_SELECT_TRACK_COMPLETE) {
        int trackIndex = event[TXVodPlayEvent.EVT_KEY_SELECT_TRACK_INDEX];
        int errorCode = event[TXVodPlayEvent.EVT_KEY_SELECT_TRACK_ERROR_CODE];
    }
});
```

#### Step 3: Monitor subtitle text content





```
// After starting to play the video, monitor the subtitle text content callback
_controller.onPlayerEventBroadcast.listen((event) async {
    if(event["event"] == TXVodPlayEvent.EVENT_SUBTITLE_DATA) {
        // Subtitle text content, can be used for display
        String subtitleDataStr = event[TXVodPlayEvent.EXTRA_SUBTITLE_DATA] ?? "";
    }
});
```

# 17. Switching between multiple audio tracks

Note: This feature requires Player Premium version 11.7 to be supported.

The advanced version of the player SDK supports switching between multiple audio tracks embedded in the video. Usage is as follows:



```
// Return the audio track information list
List<TXTrackInfo> trackInfoList = await _vodPlayerController.getAudioTrackInfo();
for (TXTrackInfo tempInfo in trackInfoList) {
    if(tempInfo.trackIndex == 0) {
      //Switch to the required audio track by judging trackIndex or name
      _vodPlayerController.selectTrack(tempInfo.trackIndex);
    } else {
```

# 🕗 Tencent Cloud

```
// Unnecessary audio tracks are deselectTrack
_vodPlayerController.deselectTrack(tempInfo.trackIndex);
}
```

# Using Advanced Features

# 1. Video preloading

### Step 1. Use video preloading

In UGSV playback scenarios, the preloading feature contributes to a smooth viewing experience: When a video is being played, you can load the next video to be played back on the backend. When a user switches to the next video, it will already be loaded and can be played back immediately.

Video preloading can deliver an instant playback effect but has certain performance overheads. If your business needs to preload many videos, we recommend you use this feature together with video predownloading.

This is how seamless switching works in video playback. You can use setAutoPlay in

TXVodPlayerController to implement the feature as follows:







```
// Play back video A: If `autoPlay` is set to `true`, the video will be immediately
String urlA = "http://1252463788.vod2.myqcloud.com/xxxxx/v.f10.mp4";
controller.setAutoPlay(isAutoPlay: true);
controller.startVodPlay(urlA);
```

```
// To preload video B when playing back video A, set `setAutoPlay` to `false`
String urlB = "http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
controller.setAutoPlay(isAutoPlay: false);
controller.startVodPlay(urlB); // The video won't be played back immediately but wi
```



After video A ends and video B is automatically or manually switched to, you can call the resume function to immediately play back video B.

#### Note:

After autoPlay is set to false, make sure that video B has been prepared before calling resume, that is, you should call it only after the PLAY\_EVT\_VOD\_PLAY\_PREPARED event of video B (2013: the player has been prepared, and the video can be played back) is detected.



```
controller.onPlayerEventBroadcast.listen((event) async {// Subscribe to status chan
if(event["event"] == TXVodPlayEvent.PLAY_EVT_PLAY_END) {
    await _controller_A.stop();
```

```
await _controller_B.resume();
}
});
```

#### Step 2. Configure the video preloading buffer

You can set a large buffer to play back videos more smoothly under unstable network conditions. You can set a smaller buffer to reduce the traffic consumption.

#### Preloading buffer size

This API is used to control the maximum buffer size before the playback starts in preloading scenarios (that is, AutoPlay of the player is set to false before video playback starts).





```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setMaxPreloadSize(2); // Maximum preloading buffer size in MB. Set it based
mVodPlayer.setConfig(config); // Pass in `config` to `mVodPlayer`
```

#### Playback buffer size

During normal video playback, you can control the maximum size of the data buffered from the network in advance. If the maximum buffer size is not configured, the player will use the default buffer policy to guarantee a smooth playback experience.





FTXVodPlayConfig config = FTXVodPlayConfig(); config.maxBufferSize = 10; // The maximum buffer size during playback in MB \_controller.setPlayConfig(config); // Pass in `config` to `controller`

## 2. Video predownloading

You can download part of the video content in advance without creating a player instance, so as to start playing back the video faster when using the player. This helps deliver a better playback experience. Before using the playback service, make sure that video cache has been set.



Sample:

### Note:

1. TXPlayerGlobalSetting is the global cache setting API, and the original TXVodConfig API has been deprecated.

2. The global cache directory and size settings have a higher priority than those configured in TXVodConfig of the player.

Download via media URL

Download via media FileId



 $//\ Set$  the global cache directory and cache size of the playback engine

# 🔗 Tencent Cloud

```
SuperPlayerPlugin.setGlobalMaxCacheSize(200);
// The cache path is set to the application's sandbox directory by default. You onl
// On Android, videos will be cached to the `Android/data/your-pkg-name/files/testC
// On iOS, videos will be cached to the `Documents/testCache` directory in the sand
SuperPlayerPlugin.setGlobalCacheFolderPath("postfixPath");
String palyrl = "http://****";
// Start predownloading
int taskId = await TXVodDownloadController.instance.startPreLoad(palyrl, 3, 1920*10
  onCompleteListener:(int taskId,String url) {
   print('taskID=${taskId} ,url=${url}');
  }, onErrorListener: (int taskId, String url, int code, String msg) {
   print('taskID=${taskId} ,url=${url}, code=${code} , msg=${msg}');
  }
);
// Cancel predownloading
TXVodDownloadController.instance.stopPreLoad(taskId);
```





```
// Set the global cache directory and cache size of the playback engine
SuperPlayerPlugin.setGlobalMaxCacheSize(200);
// The cache path is set to the application's sandbox directory by default. You onl
// On Android, videos will be cached to the `Android/data/your-pkg-name/files/testC
// On iOS, videos will be cached to the `Documents/testCache` directory in the sand
SuperPlayerPlugin.setGlobalCacheFolderPath("postfixPath");
int retTaskId = -1;
TXVodDownloadController.instance.startPreload(TXPlayInfoParams(appId: 0, fileId: "y
```

onStartListener: (taskId, fileId, url, params) {
 // TXVodDownloadController will call this block for callback taskId and video

```
retTaskId = taskId;
},
onCompleteListener: (taskId, url) {
    // preDownload complete
    },
    onErrorListener: (taskId, url, code, msg) {
        // preDownload error
    });
// Cancel predownloading
TXVodDownloadController.instance.stopPreLoad(retTaskId);
```

## 3. Video download

Video download allows users to download online videos and watch them offline. In addition, the Player SDK provides the local encryption feature, so that downloaded local files are still encrypted and can be decrypted and played back only in the specified player. This feature can effectively prevent downloaded videos from being distributed without authorization and protect the video security.

As HLS streaming media cannot be directly saved locally, you cannot download them and play back them as local files. You can use the video download scheme based on TXVodDownloadController to implement offline HLS playback.

#### Note:

Currently, TXVodDownloadController can cache only non-nested HLS files but not MP4 and FLV files. The Player SDK already supports playing back local MP4 and FLV files.

#### Step 1. Make preparations

TXVodDownloadController is designed as a singleton; therefore, you cannot create multiple download objects. It is used as follows:





// The cache path is set to the application's sandbox directory by default. You onl // On Android, videos will be cached to the `Android/data/your-pkg-name/files/testC // On iOS, videos will be cached to the `Documents/testCache` directory in the sand SuperPlayerPlugin.setGlobalCacheFolderPath("postfixPath");

#### Step 2. Start the download

You can start the download through Fileid or URL as follows: Through Fileid Through URL For download through Fileid , you need to pass in AppID , Fileid , and qualityId at least. For signed videos, you also need to pass in pSign . If no specific value is passed in to userName , it will be default by default.

#### Note:

You can download encrypted videos only through Fileid and must enter the psign parameter.



| QUALITY_240P | 240p   |
|--------------|--|
| QUALITY_360P | 360P   |
| QUALITY_480P | 480p   |
| QUALITY_540P | 540p   |
|              | QUALITY_240P<br>QUALITY_360P<br>QUALITY_480P<br>QUALITY_540P |

// QUALITY\_720P 720p // QUALITY\_1080P 1080p // QUALITY\_2K 2k // QUALITY\_4K 4k // The quality parameter can be customized to take the minimum value of the resolut // (for example, for a resolution of 1280\*720, if you want to download a stream of // you can pass in QUALITY\_720P for the quality parameter). The player SDK will sel // a resolution less than or equal to the passed-in resolution for downloading. // Using quality ID to download DownloadHelper.instance.startDownload(videoModel, qualityId); // Using mediaInfo to download DownloadHelper.instance.startDownloadOrg(mediaInfo);

You need to pass in the download URL at least. Only the non-nested HLS, i.e., single-bitstream HLS, is supported. If no specific value is passed in to userName, it will be default by default.





TXVodDownloadMedialnfo medialnfo = TXVodDownloadMedialnfo(); medialnfo.url = "http://1500005830.vod2.myqcloud.com/43843ec0vodtranscq1500005830/0 TXVodDownloadController.instance.startDonwload(medialnfo);

#### Step 3. Receive the task information

Before receiving the task information, you need to set the callback listener first.





```
TXVodDownloadController.instance.setDownloadObserver((event, info) {
```

```
}, (errorCode, errorMsg, info) {
```

});

You may receive the following task events:

| Event                | Description  |
|----------------------|--|
| EVENT_DOWNLOAD_START | The task started, that is, the SDK started the download. |

| EVENT_DOWNLOAD_PROGRESS | The task progress. During download, the SDK will frequently call back this API. You can use <pre>mediaInfo.getProgress()</pre> to get the current progress. |
|-------------------------|---|
| EVENT_DOWNLOAD_STOP     | The task stopped. When you call stopDownload to stop the download, if this message is received, the download is stopped successfully.                       |
| EVENT_DOWNLOAD_FINISH   | Download was completed. If this callback is received, the entire file has been downloaded, and the downloaded file can be played back by TXVodPlayer .      |

If the downlodOnErrorListener method is called back, a download error occurred. If the network is disconnected during download, this API will be called back and the download task will stop. As the downloader can download multiple files at a time, the callback API carries the

TXVodDownloadMedialnfo object. You can access the URL or dataSource to determine the download source and get other information such as download progress and file size.

#### Step 4. Stop the download

You can call the TXVodDownloadController.instance.stopDownload() method to stop the download. The parameter is the TXVodDownloadMedialnfo object passed in when download starts. **The SDK supports checkpoint restart.** If the download directory is not changed, when you resume downloading a file, the download will start from the point where it stopped.

#### Step 5. Manage downloads

You can get the download lists of all accounts or the specified account.





// Get the download lists of all users // You can distinguish between the download lists of different users by `userName` List<TXVodDownloadMedialnfo> downloadInfoList = await TXVodDownloadController.insta

To get the download information of a Fileid , such as the current download status and progress, you need to pass in AppID , Fileid , and qualityId .





// Get the download information of a video
TXVodDownloadMedialnfo downloadInfo = await TXVodDownloadController.instance.getDow
int? duration = downloadInfo.duration; // Get the total duration
int? playableDuration = downloadInfo.playableDuration; // Get the playable duration
double? progress = downloadInfo.progress; // Get the download progress
String? playPath = downloadInfo.playPath; // Get the offline playback path, which c
int? downloadState = downloadInfo.downloadState; // Get the download status. For mo





// Delete the download information
bool result = await TXVodDownloadController.instance.deleteDownloadMediaInfo(medial

## 4. Encrypted playback

The video encryption solution is used in scenarios where the video copyright needs to be protected, such as online education. To encrypt your video resources, you need to alter the player and encrypt and transcode video sources. For more information, see Media Encryption and Copyright Protection Overview.



After you get the appId as well as the encrypted video's fileId and psign in the Tencent Cloud console, you can play back the video as follows:



# 5. Player configuration

Before calling statPlay , you can call setConfig to configure the player parameters, such as player connection timeout period, progress callback interval, and maximum number of cached files. TXVodPlayConfig allows you to configure detailed parameters. For more information, see Basic Configuration API. Below is the configuration sample code:



```
FTXVodPlayConfig config = FTXVodPlayConfig();
// If `preferredResolution` is not configured, the 720x1280 resolution stream will
config.preferredResolution = 720 * 1280;
config.enableAccurateSeek = true; // Set whether to enable accurate seek. Default
config.progressInterval = 200; // Set the progress callback interval in millisecon
config.maxBufferSize = 50; // Set the maximum preloading buffer size in MB
```

#### \_controller.setPlayConfig(config);

#### Specifying resolution when playback starts

When playing back an HLS multi-bitrate video source, if you know the video stream resolution information in advance, you can specify the preferred resolution before playback starts, and the player will select and play back the stream at or below the preferred resolution. In this way, after playback starts, you don't need to call setBitrateIndex to switch to the required bitstream.



```
FTXVodPlayConfig config = FTXVodPlayConfig();
// The parameter passed in is the product of the video width and height. You can p
```

# Stencent Cloud

```
config.preferredResolution = 720 * 1280;
_controller.setPlayConfig(config);
```

#### Specify media type before broadcasting

When the type of media asset to be played is known in advance, the playback speed can be enhanced by configuring TXVodPlayConfig#mediaType to reduce the internal playback type detection of the player SDK.



```
FTXVodPlayConfig config = FTXVodPlayConfig();
config.mediaType = TXVodPlayEvent.MEDIA_TYPE_FILE_VOD; // Used to improve MP4 pla
config.mediaType = TXVodPlayEvent.MEDIA_TYPE_HLS_VOD; // Used to improve HLS pla
_controller.setPlayConfig(config);
```



#### Setting playback progress callback interval



```
FTXVodPlayConfig config = FTXVodPlayConfig();
config.progressInterval = 200; // Set the progress callback interval in millisecon
_controller.setPlayConfig(config);
```

**Player Event Listening** 

You can listen on the player's playback events through onPlayerEventBroadcast of

TXVodPlayerController to sync information to your application.

# Playback event notifications ( onPlayerEventBroadcast )

| Event ID                   | Code | Description  |
|----------------------------|------|--|
| PLAY_EVT_PLAY_BEGIN        | 2004 | Video playback started.  |
| PLAY_EVT_PLAY_PROGRESS     | 2005 | The video playback progress (including the current playback progress, loading progress, and total video duration). |
| PLAY_EVT_PLAY_LOADING      | 2007 | The video is being loaded. The LOADING_END event will be reported if video playback resumes.                       |
| PLAY_EVT_VOD_LOADING_END   | 2014 | Video loading ended, and video playback resumed.   |
| VOD_PLAY_EVT_SEEK_COMPLETE | 2019 | Seeking was completed. The seeking feature is supported by v10.3 or later.   |

#### Stop events

| Event ID                | Code  | Description  |
|-------------------------|-------|--|
| PLAY_EVT_PLAY_END       | 2006  | Video playback ended.  |
| PLAY_ERR_NET_DISCONNECT | -2301 | The network was disconnected and could not be reconnected after multiple retries. You can restart the player to perform more connection retries. |
| PLAY_ERR_HLS_KEY        | -2305 | Failed to get the HLS decryption key.  |

#### Warning events

You can ignore the following events, which are only used to notify you of some internal events of the SDK.

| Event ID                       | Code | Description  |
|--------------------------------|------|--|
| PLAY_WARNING_VIDEO_DECODE_FAIL | 2101 | Failed to decode the current video frame.  |
| PLAY_WARNING_AUDIO_DECODE_FAIL | 2102 | Failed to decode the current audio frame.  |
| PLAY_WARNING_RECONNECT         | 2103 | The network was disconnected, and automatic reconnection was performed (the PLAY_ERR_NET_DISCONNECT event will be thrown after three failed attempts). |
| PLAY_WARNING_HW_ACCELERATION_FAIL | 2106 | Failed to start the hardware decoder, and the |
|-----------------------------------|------|---|
|                                   |      | software decoder was used instead.            |

#### **Connection events**

The following server connection events are mainly used to measure and collect the server connection time:

| Event ID                   | Code | Description  |
|----------------------------|------|--|
| PLAY_EVT_VOD_PLAY_PREPARED | 2013 | The player has been prepared and can start playback. If<br>autoPlay is set to false , you need to call<br>resume after receiving this event to start playback. |
| PLAY_EVT_RCV_FIRST_I_FRAME | 2003 | The network received the first renderable video data packet (IDR).   |

#### Image quality events

The following events are used to get image change information:

| Event ID                   | Code | Description                   |
|----------------------------|------|-------------------------------|
| PLAY_EVT_CHANGE_RESOLUTION | 2009 | The video resolution changed. |
| PLAY_EVT_CHANGE_ROTATION   | 2011 | The MP4 video was rotated.    |

#### Video information events

| Event ID                   | Code | Description  |
|----------------------------|------|--|
| PLAY_EVT_GET_PLAYINFO_SUCC | 2010 | Obtained the information of the file played back successfully. |

If you play back a video through fileId and the playback request succeeds (called API:

startVodPlay(TXPlayerAuthBuilder authBuilder) ), the SDK will notify the upper layer of some request information, and you can parse param to get the video information after receiving the

TXLiveConstants.PLAY\_EVT\_GET\_PLAYINFO\_SUCC event.

| Video Information  | Description            |
|--------------------|------------------------|
| EVT_PLAY_COVER_URL | Video thumbnail URL    |
| EVT_PLAY_URL       | Video playback address |
| EVT_PLAY_DURATION  | Video duration         |
|                    |                        |

| EVT_TIME                     | Event occurrence time   |
|------------------------------|---|
| EVT_UTC_TIME                 | UTC time  |
| EVT_DESCRIPTION              | Event description   |
| EVT_PLAY_NAME                | Video name  |
| EVT_IMAGESPRIT_WEBVTTURL     | The download URL of the image sprite WebVTT file, which is supported by v10.2 or later. |
| EVT_IMAGESPRIT_IMAGEURL_LIST | The download URL of the image sprite image, which is supported by v10.2 or later.       |
| EVT_DRM_TYPE                 | The encryption type, which is supported by v10.2 or later.                              |

Below is the sample code for using onPlayerEventBroadcast to get the video playback information:





```
_controller.onPlayerEventBroadcast.listen((event) async {
    if (event["event"] == TXVodPlayEvent.PLAY_EVT_PLAY_BEGIN || event["event"] == TXV
    // code ...
    } else if (event["event"] == TXVodPlayEvent.PLAY_EVT_PLAY_PROGRESS) {
    // code ...
    }
});
```

Playback status feedback ( onPlayerNetStatusBroadcast )

The status feedback is triggered once every 0.5 seconds to provide real-time feedback on the current status of the pusher. It can act as a dashboard to inform you of what is happening inside the SDK so you can better understand the current video playback status.

| Parameter                | Description   |
|--------------------------|---|
| NET_STATUS_CPU_USAGE     | Current instantaneous CPU utilization   |
| NET_STATUS_VIDEO_WIDTH   | Video resolution - width  |
| NET_STATUS_VIDEO_HEIGHT  | Video resolution - height   |
| NET_STATUS_NET_SPEED     | Current network data reception speed  |
| NET_STATUS_VIDEO_FPS     | Current video frame rate of streaming media   |
| NET_STATUS_VIDEO_BITRATE | Current video bitrate in Kbps of streaming media                                      |
| NET_STATUS_AUDIO_BITRATE | Current audio bitrate in Kbps of streaming media                                      |
| NET_STATUS_VIDEO_CACHE   | Buffer (`jitterbuffer`) size. If the current buffer length is 0, lag will occur soon. |
| NET_STATUS_SERVER_IP     | Connected server IP   |

Below is the sample code of using onNetStatus to get the video playback information:





```
_controller.onPlayerNetStatusBroadcast.listen((event) async {
    int videoWidth = event[TXVodNetEvent.NET_STATUS_VIDEO_WIDTH];
});
```

## Video playback status feedback

The video playback status will be notified every time the playback status changes.

The enumeration class TXPlayerState is used to transfer the event.

Status

Description

| paused    | The playback was paused  |
|-----------|--------------------------|
| failed    | The playback failed      |
| buffering | Buffering                |
| playing   | Playing back             |
| stopped   | The playback was stopped |
| disposed  | The control was released |

Below is the sample code for using <code>onPlayerState</code> to get the video playback status:





\_controller.onPlayerState.listen((val) { });

### System volume level listening

To help you monitor the video playback volume level, the SDK encapsulates the volume level change notification into an event at the Flutter layer. You can directly use SuperPlayerPlugin to listen on the volume level change of the current device.

Below is the sample code for using onEventBroadcast to get the volume level status of the device:





```
SuperPlayerPlugin.instance.onEventBroadcast.listen((event) {
    int eventCode = event["event"];
});
```

#### The relevant events are as described below:

| Status                  | Code | Description  |
|-------------------------|------|--|
| EVENT_VOLUME_CHANGED    | 1    | The volume level changed.                                  |
| EVENT_AUDIO_FOCUS_PAUSE | 2    | The volume level output and playback focus were lost. This |



|                        |   | event is applicable only to Android.   |
|------------------------|---|--|
| EVENT_AUDIO_FOCUS_PLAY | 3 | Obtained the volume level output focus successfully. This event is applicable only to Android. |

# **PiP event listening**

As PiP used by the SDK is based on the PiP capabilities of the system, after you enter the PiP mode, a series of notifications are provided to help you adjust the UI accordingly.

| Status                          | Code | Description   |
|---------------------------------|------|---|
| EVENT_PIP_MODE_ALREADY_ENTER    | 1    | The player has entered the PiP mode.  |
| EVENT_PIP_MODE_ALREADY_EXIT     | 2    | The player has exited the PiP mode.   |
| EVENT_PIP_MODE_REQUEST_START    | 3    | The player requests to enter the PiP mode.                                      |
| EVENT_PIP_MODE_UI_STATE_CHANGED | 4    | The PiP UI status changed. This event takes effect only on Android 31 or later. |
| EVENT_IOS_PIP_MODE_RESTORE_UI   | 5    | The UI was reset. This event takes effect only on iOS.                          |
| EVENT_IOS_PIP_MODE_WILL_EXIT    | 6    | The player will exit the PiP mode. This event takes effect only on iOS.         |

Below is the sample code for using onExtraEventBroadcast to listen on PiP events:





```
SuperPlayerPlugin.instance.onExtraEventBroadcast.listen((event) {
    int eventCode = event["event"];
});
```

# Advanced Features Web Advanced Features Security Check Plugin (TCPlayerSafeCheckPlugin)

Last updated : 2024-04-11 16:18:08

The TCPlayerSafeCheckPlugin is used to check if the playback environment and status are normal, ensuring playback safety. It should be used in conjunction with TCPlayer.

# **Use Conditions**

Currently, the Web Player SDK version 5.0.0 and above support the use of the VR playback plugin. VR playback requires access to Player Premium Version License (Web) for use. In the process of long-term maintenance of the player, various types of attacks have been encountered. Against behaviors where video resources can be stolen using third-party tools, this plugin has implemented targeted prevention in the following three aspects:

#### **1. MSE Environment Detection**

Some browser plugins or scripts can hijack the current playback environment by modifying the Media Source Extensions API (MSE) to intercept playback data, ultimately enabling video download. This plugin can detect and prevent such attacks.

#### 2. Security Structure Inspection

Third-party tools or scripts can modify the player's structure, removing playback markers, watermarks, etc., and enabling screen recording. This plugin monitors whether the player's structure has been tampered with. If such behavior is detected, playback is immediately halted.

### 3. API Response Integrity Verification

During the use of the player, it is necessary to interact with the Video on Demand server. If the interface data is modified, it will affect the normal playback behavior. This plugin can detect and prevent such types of attacks.

# Use Method

For integration with TCPlayer, refer to TCPlayer Integration Guide and API Documentation.

When creating a player instance, you can enable security check capabilities by claiming plugins. Once enabled, the player will automatically load and use this plugin:



```
const player = TCPlayer('player-container-id', {
   plugins: {
     SafeCheck: true,
   }
});
```



# Effect

After enabling the plugin, the player will automatically check if the environment is safe. If an attack is detected, playback will be automatically terminated, and a corresponding prompt will be displayed, as shown below:



The error codes prompted by the plugin are as follows:

| Error code | Description                         |
|------------|-------------------------------------|
| 60         | Security structure check exception  |
| 61         | API response integrity exception    |
| 62         | MSE environment detection exception |

# VR Playback Plugin (TCPlayerVRPlugin)

Last updated : 2024-04-11 16:18:08

The TCPlayerVRPlugin can be used for VR panoramic video playback. During playback, you can change the viewing angle through gyroscope rotation or gesture operations. It offers various properties and methods to control playback performance and supports both PC and mobile platforms.

### **Use Conditions**

Currently, the Web Player SDK version 5.0.0 and above support the use of the VR playback plugin. VR playback requires access to Player Premium Version License (Web) for use.

### **Connection Method**

For the player initialization process, see TCPlayer Integration Guide and API Documentation.

When initializing the player instance, you can enable VR playback by claiming plugins. Once enabled, the player will automatically load and use this plugin:





```
const player = TCPlayer('player-container-id', { // player-container-id is the play
    plugins: {
        VR: {
            isEnableController: true,
            ...
        },
        },
    });
```

## **VR Plugin Property Description**



| Name               | Description  | Default Value |
|--------------------|--|---------------|
| isEnableController | Enable VR Controller                                 | true          |
| isEnableZoom       | Allow Image Scaling                                  | true          |
| yaw                | Initialize left and right viewing angles, in degrees | 0             |
| pitch              | Initialize vertical viewing angles, in degrees       | 0             |
| fov                | Initialize the field of view, in degrees             | 65            |
| yawRange           | Limit the range of view movement                     | [-180, 180]   |
| pitchRange         | Limit the range of view movement                     | [-90, 90]     |
| fovRange           | Limit the range of view movement                     | [30, 110]     |

## **VR Plugin Method Description**

After the VR plugin is initialized, it generates an instance. After instantiation, it enters the VR pattern to play videos. The VR instance can be found on the player instance, and related methods can be called through the VR instance:

#### - lookAt

Move to a specific angle of view through animation over a period of time.





player.plugins['VR'].lookAt({ yaw: 30 }, 1000);

#### - setGyroMode

If your device has motion sensors (gyroscope, accelerometer), you can change the viewing angle through the device's motion. This method can be set to 'VR' | 'none' | 'yawPitch'.





player.plugins['VR'].setGyroMode('none');

#### - enableSensor

Access permission to use motion sensors. Typically, on Android devices, motion sensors are enabled by default, while on iOS 13+, manual access permission is required through user interaction.





player.plugins['VR'].enableSensor();

#### Note:

- 1. In a browser hijacking environment, it is impossible to support the playback of VR videos.
- 2. After initialization, the Android player will default to the VR pattern and activate the gyroscope.
- 3. On the iOS side, performance may vary depending on the system version:
- 3.1 For system versions 13+, you need to manually click on the page to trigger user interaction and access permissions before the gyroscope can be activated.

3.2 For system versions 12.2 to 13, you need to go into the system Settings to manually enable the motion sensor. The usual path to do this is Settings > Safari > Motion & Orientation Access. After enabling the sensor, refresh the page to activate it.

## Sample code

Click here to see the sample code.

# Advanced Mobile Features Picture-in-Picture Component (TUIPIP) iOS

Last updated : 2024-04-17 11:49:14

# Introduction

### Application scenarios

Advanced Picture-in-Picture (PIP) is an upgraded version of the basic PIP, mainly used for encrypted video PIP, offline playback PIP, and seamless switching from foreground to PIP scenes. It optimizes the implementation and logic, and achieves a true "instant switch" effect without long waiting times.

Advantages of Advanced PIP:

Encrypted video PIP: integrated with the existing player for encrypted playback, realizing PIP playback based on encrypted templates without the need to switch player types.

Offline playback PIP: supports local video PIP playback, including ordinary videos, encrypted videos, etc. "Instant switch" effect: no need to click the PIP button to switch, just go to the background and PIP will start immediately, achieving a true "instant switch".

# Requirements

System version:  $iOS \ge 14.0$ ,  $iPad \ge 9.0$ . Hardware devices: iPhone 8 and above. **SDK version: 11.4 or above.** 

# **Integration Steps**

### Upgrade SDK version and configure resources

### 1. Upgrade SDK version:

Advanced Picture-in-Picture requires the use of the SDK. Before using the advanced Picture-in-Picture version, the SDK version needs to be upgraded to version 11.3 or above for the advanced version, or version 11.4 or above for the basic version, otherwise it cannot be used. At the same time, the basic and advanced versions of Picture-in-Picture

can coexist with compatibility and no functional conflicts. If you want to upgrade the SDK version, please refer to the SDK integration guide.

#### 2. Introduce bundle resources

Because the SDK needs to use resources from TXVodPlayer.bundle, the bundle file needs to be downloaded and introduced into the project before compilation. Please do not modify the bundle or the names of the resources used inside, as doing so may cause seamless switching of picture-in-picture to fail.



### 3. Activate the premium version license for the player

The advanced picture-in-picture version requires the mobile player premium version license. You can refer to the mobile player license guide to obtain it. If you have obtained the corresponding License, you can go to Tencent Cloud Cube Console > License Management > Mobile License to obtain the corresponding LicenseURL and LicenseKey. If you do not apply for the Player Advanced Package License, entering Picture-in-Picture will be invalid. After obtaining the License information, before calling the relevant interface of the SDK, initialize the License through the following interface. It is recommended to make the following settings in - [AppDelegate application:didFinishLaunchingWithOptions:]:





```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
NSString * const licenceURL = @"<your license Url>";
NSString * const licenceKey = @"<your license key>";
//TXLiveBase is in "TXLiveBase.h"
[TXLiveBase setLicenceURL:licenceURL key:licenceKey];
NSLog(@"SDK Version = %@", [TXLiveBase getSDKVersionStr]);
}
```

#### Quick access to picture-in-picture function

#### 1. Permission granted

Picture-in-picture (PictureInPicture) has been launched in iOS 9, but it was only available on the iPad before. To use picture-in-picture, the iPhone needs to be updated to iOS 14. Currently, Tencent Cloud Player can support in-app and out-of-app picture-in-picture capabilities, which greatly meets the needs of users. You need to enable background mode before use. The steps are: Select the corresponding Target > Signing & Capabilities > Background Modes in XCode and check "Audio, AirPlay, and Picture in Picture".

| Background Modes |   |
|------------------|---|
|                  | Modes 🗸 Audio, AirPlay, and Picture in Picture<br>Location updates<br>Voice over IP |

#### 2. Set configuration options

In order to use the automatic picture-in-picture function, you need to turn on the automatic picture-in-picture button in the settings. The specific path is to select on iPhone or iPad: Settings > General > Picture-in-Picture > Automatically turn on Picture-in-Picture, select to turn it on.



#### 3. Set proxy

In order to facilitate monitoring of the picture-in-picture status, vodDelegate needs to be set to implement the picturein-picture related callbacks in TXVodPlayListener. You can perform related business operations based on various



status and error messages in the callback, such as continuing to play, pausing or exiting picture-in-picture, etc.



```
/**
 * Picture-in-picture status callback
 *
 */
- (void)onPlayer:(TXVodPlayer *)player pictureInPictureStateDidChange:(TX_VOD_PLAYE
 /**
 * Picture-in-picture error message callback
 *
```

\*/

- (void) onPlayer: (TXVodPlayer \*) player pictureInPictureErrorDidOccur: (TX\_VOD\_PLAYER

#### 4. Code example using picture-in-picture capability

#### Note:

To use the automatic picture-in-picture function, make sure the player is in the playing state. If the player is paused or stopped, the automatic picture-in-picture function cannot be used.

The isSupportSeamlessPictureInPicture interface can only be used after the application loads the advanced version License. At the same time, this interface can only determine whether the device itself supports automatic switching of picture-in-picture. Due to system limitations, it cannot determine the user's setting permission for automatic picture-in-picture, and needs to be guided by itself.

Before playing, set whether to allow "automatic switching of picture-in-picture function".





```
// 1. Before playing, set whether the "automatic switching Picture-In-Picture funct
// YES means allowed, NO means not allowed, the default is NO
[TXVodPlayer setPictureInPictureSeamlessEnabled:YES];
```

```
// 2. Enter picture-in-picture
if (![TXVodPlayer isSupportPictureInPicture]) {
    //The device does not support picture-in-picture and exit directly.
    return;
}
```

# TUIPlayerShortVideo iOS

Last updated : 2024-06-17 17:06:27

# **Component Introduction**

TUIPlayerShortVideo component is a short video component launched by Tencent Cloud with excellent performance, supporting ultra-fast first frame and smooth sliding of videos, and providing high-quality playback experience. First frame opens in seconds: First frame time is one of the core indicators of short video applications, which directly affects the user's viewing experience. Short video components use technologies such as pre-playback, pre-download, player reuse and precise traffic control to achieve a high-quality playback experience with extremely fast first frame and smooth sliding, thereby increasing user playback volume and retention time.

Excellent performance: Through the optimization of player reuse and loading strategies, memory and CPU consumption are always kept at a low level while ensuring excellent smoothness.

Quick integration: The component encapsulates complex playback operations, provides a default playback UI, and supports both FileId and URL playback, which can be quickly integrated into your project at low cost.

# Effect comparison

From the example video below you can see the difference before and after implementing the best short video strategy. There is an obvious first frame lag before optimization.

After optimization, the playback is smooth and the average playback time after optimization reaches 10 milliseconds - 30 milliseconds.

| Not optimized for short videos | Optimized short video |
|--------------------------------|-----------------------|
|                                |                       |
|                                |                       |
|                                |                       |
|                                |                       |
|                                |                       |

# TUIPlayerKit Download

TUIPlayerKit SDK and Demo can be downloaded by clicking here.

# Integration Guide

## 1. Dependencies

The SDKs that TUIPlayerShortVideo depends on are : TUIPlayerCore TXLiteAVSDK ≥ 11.4 SDWebImage Masonry

### 2. Environmental requirements

System version  $\therefore \ge iOS 9.0$ Development Environment  $\therefore \ge Xcode 14.0$  (It is recommended to use the latest version)

### 3. Integrated TUIPlayerCore

Unzip the downloaded TUIPlayerKit resource package, add the TUIPlayerCore.xcframework component SDK to the appropriate location of the Xcode Project in your project, select the appropriate target, and check Do Not Embed.

### 4. Integrated TUIPlayerShortVideo

Unzip the downloaded TUIPlayerKit resource package, add the TUIPlayerShortVideo.xcframework component SDK to the appropriate location of your Xcode Project, select the appropriate target, and check Do Not Embed.

### 5. Integrated TXLiteAVSDK

For TXLiteAVSDK integration methods, please refer to TXLiteAVSDK Integration Guide.

### 6. Integrated SDWebImage

For downloading and integrating SDWebImage, please refer to the GitHub instructions.

### 7. Integrated Masonry

For downloading and integrating Masonry, please refer to the GitHub instructions.

### 8. Pod Integration

If your project supports pod, you can also integrate it through the spec file we provide, as follows:





```
pod 'TUIPlayerShortVideo' ,:path => '../../.SDK/TUIPlayerShortVideoSDK/'
pod 'TUIPlayerCore' ,:path => '../../SDK/TUIPlayerCoreSDK/'
```

#### Attention:

Please configure Path according to your own project file path.

Remote Pod integration is not currently supported.

# Interface Instructions

## 1. Quick access

### 1.1. Configure Player Premium License

The use of TUIPlayer Kit components requires the use of the Mobile Player Premium License, which you can obtain by referring to the Mobile Player License Guide. If you have already obtained the corresponding license, you can go to Tencent Cloud Visual Cube Console > License Management > Mobile License to obtain the corresponding LicenseURL and LicenseKey. If you do not apply for the Mobile Player Premium License, video playback failures, black screens, etc. will occur.

Before calling related functions, configure Licence in your project. It is recommended to reference the TUIPlayerCore module in \_ [AppDelegate application:didFinishLaunchingWithOptions:] make the following configuration:



```
NSString * const licenceURL = @"<Obtained licenseUrl>";
NSString * const licenceKey = @"<The key obtained>";
[TXLiveBase setLicenceURL:licenceUrl key:licenceKey];
[[TXLiveBase sharedInstance] setDelegate:self];
```

## 1.2. Play

Initialize TUIShortVideoView, as follows:





```
- (TUIShortVideoView *)videoView {
```

```
if (!_videoView) {
```

```
///Setting up a custom UI
TUIPlayerShortVideoUIManager *uiManager = [[TUIPlayerShortVideoUIManager al
[uiManager setControlViewClass: TUIPlayerShortVideoControlView.class];
[uiManager setControlViewClass: TUIPSControlLiveView.class viewType:TUI_ITE
[uiManager setControlViewClass: TUIPSControlCustomView.class viewType:TUI_I
[uiManager setLoadingView:[[TUIPSDLoadingView alloc] init]];
```



```
__videoView = [[TUIShortVideoView alloc] initWithUIManager:uiManager];
__videoView.delegate = self;
__videoView.customCallbackDelegate = self;
//_videoView.isAutoPlay = NO;
// Set your playback strategy
TUIPlayerVodStrategyModel *model = [[TUIPlayerVodStrategyModel alloc] init]
model.mPreloadConcurrentCount = 1;
model.preDownloadSize = 1;
model.enableAutoBitrate = NO;
// live strategy
TUIPlayerLiveStrateyModel *liveStrateyModel = [[TUIPlayerLiveStrateyModel a
[_videoView setShortVideoLiveStrategyModel:liveStrateyModel];
}
return _videoView;
}
```

Add an instance of TUIShortVideoView to the View you want to present, as shown in the following code:



videoView.frame = self.view.bounds;
[self.view addSubview:self.videoView];

Then add your videos array:




```
TUIPlayerVideoModel *model1 = [[TUIPlayerVideoModel alloc] init]; ///Video Data
TUIPlayerLiveModel *model2 = [[TUIPlayerLiveModel alloc] init]; ///Live data
TUIPlayerDataModel *model3 = [[TUIPlayerDataModel alloc] init]; ///Custom Data
/// Here, you can decide the amount of data per page based on your business situati
NSArray *videos1 = @[model1,model2,model3];
[self.videoView setShortVideoModels:videos1];
```

After the first set of videos is played, you need to continue to insert your second set of video data in the TUIShortVideoViewDelegate delegate method:





```
TUIPlayerVideoModel *model1 = [[TUIPlayerVideoModel alloc] init]; ///Video Data
TUIPlayerLiveModel *model2 = [[TUIPlayerLiveModel alloc] init]; ///Live data
TUIPlayerDataModel *model3 = [[TUIPlayerDataModel alloc] init]; ///Custom Data
/// Here, you can decide the amount of data per page based on your business situati
NSArray *videos2 = @[model1,model2,model3];
-(void)onReachLast {
    ///Here you can make a data index record and continue to insert your 3rd 4th 5t
    [self.videoView appendShortVideoModels:videos2];
}
```

# 1.3. TUIShortVideoView

The main interfaces of TUIShortVideoView are as follows:

| Parameter name                 | Implication   |
|--------------------------------|---|
| isAutoPlay                     | Whether to automatically play the first video when loading for the first time, default is YES |
| videos                         | Read-only property, get the data currently in the video list                                  |
| currentVideoModel              | The video model currently playing   |
| currentVideoIndex              | Index of the video currently playing  |
| currentPlayerStatus            | The current player's playback status  |
| isPlaying                      | Is the current player playing   |
| delegate                       | delegate  |
| refreshControl                 | Set the pull-down refresh control   |
| initWithUIManager              | Initialization  |
| setShortVideoStrategyModel     | Set the live broadcast strategy   |
| setShortVideoLiveStrategyModel | Set the live broadcast strategy   |
| setShortVideoModels            | Setting up a data source for the first time   |
| appendShortVideoModels         | Add video data source   |
| removeAllVideoModels           | Delete all video data   |
| setPlaymode                    | Video playback mode, single loop or list loop, the former is the default                      |
| pause                          | Pause   |
| resume                         | Resume  |
| destoryPlayer                  | Destroy the player  |
| didScrollToCellWithIndex       | Jump to the video with the specified index  |
| startLoading                   | Display loading image   |
| stopLoading                    | Hide loading icon   |
| currentPlayerSupportedBitrates | The bitrate supported by the currently playing video  |
|                                |   |

| bitrateIndex            | Get the bitrate index of the current playback |
|-------------------------|---|
| switchResolution:index: | Switch resolution                             |
| pausePreload            | Pause preloading                              |
| resumePreload           | Resume preload                                |
| getDataManager          | Get Data Manager                              |
| getVodStrategyManager   | Get the On-Demand Policy Manager              |
| getLiveStrategyManager  | Get the Live Strategy Manager                 |

# 2. Global Configuration

You can set some global configurations in TUIPlayerCore via the TUIPlayerConfig model.

The main parameters of TUIPlayerConfig are shown in the table below:

| Parameter name | Implication                                       |
|----------------|---|
| enableLog      | Whether to allow printing logs, the default is NO |

Then configure globally through TUIPlayerCore:





```
TUIPlayerConfig *config = [TUIPlayerConfig new];
config.enableLog = YES;
[[TUIPlayerCore shareInstance] setPlayerConfig:config];
```

# 3. Player policy configuration

## 3.1. VOD Playback Policy Settings

You can use the TUIPlayerVodStrategyModel model to configure the on-demand playback strategy.

The main parameters of TUIPlayerVodStrategyModel are shown in the following table:

| Parameter name          | Implication  |
|-------------------------|--|
| mPreloadConcurrentCount | The number of video caches, default is 3   |
| mPreloadBufferSizeInMB  | Pre-play size, in MB, default 0.5MB  |
| preferredResolution     | Preferred resolution, default 720 * 1280   |
| progressInterval        | The progress bar callback interval, in milliseconds, default is 500ms  |
| renderMode              | Canvas fill style, the default image adapts to the screen to keep the picture intact   |
| extInfoMap              | Additional parameters, reserved  |
| enableAutoBitrate       | Whether to enable adaptive bitrate, default is NO  |
| mediaType               | Set the media type   |
| maxBufferSize           | Maximum preload size, in MB, default 10MB, this setting will affect playableDuration, the larger the setting, the more pre-cached  |
| mResumeModel            | Resume mode, default value is TUI_RESUM_MODEL_NONE   |
| preDownloadSize         | Pre-download size, in MB, default 1MB  |
| enableAccurateSeek      | Whether to seek accurately, the default is YES. After turning on accurate seek, the seek time will be 200ms longer on average  |
| audioNormalization      | Volume balance. Loudness range: -70~0 (LUFS). This configuration<br>requires LiteAVSDK 11.7 and above.<br>The following constants are for reference:<br>Off: AUDIO_NORMALIZATION_OFF (TXVodPlayConfig.h)<br>On (standard loudness): AUDIO_NORMALIZATION_STANDARD<br>(TXVodPlayConfig.h)<br>On (low loudness): AUDIO_NORMALIZATION_LOW<br>(TXVodPlayConfig.h)<br>On (high loudness): AUDIO_NORMALIZATION_HIGH<br>(TXVodPlayConfig.h)<br>The default value is AUDIO_NORMALIZATION_OFF. |
| isLastPrePlay           | Whether to keep the last pre-play, the default is NO   |
| subtitleRenderModel     | Subtitle style   |

Then configure the player strategy :





```
TUIPlayerStrategyModel *model = [[TUIPlayerStrategyModel alloc] init];
model.mPreloadConcurrentCount = 1;
model.preDownloadSize = 1;
model.enableAutoBitrate = NO;
model.mRenderMode = TUI_RENDER_MODE_FILL_SCREEN;
model.mResumeModel = TUI_RESUM_MODEL_LAST;
[_videoView setShortVideoStrategyModel:model];
```

#### 3.2. Live broadcast strategy settings

You can use the TUIPlayerLiveStrategyModel model to configure the on-demand playback strategy.

The main parameters of TUIPlayerLiveStrategyModel are shown in the following table:

| Parameter name         | Implication  |
|------------------------|--|
| isLastPrePlay          | Whether to keep the last pre-play, the default is NO   |
| mRenderMode            | Canvas fill style, default V2TXLiveFillModeFill  |
| enablePictureInPicture | YES: Enable the PIP function; NO: Disable the PIP function. Default value: NO.   |
| volume                 | The volume of the player, ranging from 0 to 100. Default value: 100.   |
| maxAutoAdjustCacheTime | The maximum time for automatic adjustment of the player cache, in seconds. The value must be greater than 0. The default value is 5. |
| minAutoAdjustCacheTime | The minimum time for automatic adjustment of the player cache, in seconds. The value must be greater than 0. The default value is 1. |
| isShowDebugView        | Whether to display the debug overlay of player status information. Default value: NO.  |

Then configure the player strategy :





TUIPlayerLiveStrategyModel \*liveStrategyModel = [[TUIPlayerLiveStrategyModel alloc]
[\_videoView setShortVideoLiveStrategyModel:liveStrategyModel];

#### 3.3. Dynamic policy adjustment

Both on-demand and live broadcast strategies support dynamic adjustment. The steps are as follows: :

1、Get the live broadcast & on-demand strategy management class through TUIShortVideoView.





TUIPlayerVodStrategyManager \*VodStrategyManager = [\_videoView getVodStrategyManager TUIPlayerVodStrategyManager \*LiveStrategyManager = [\_videoView getLiveStrategyManag

2、Adjust playback strategies through VodStrategyManager and LiveStrategyManager.





[VodStrategyManager setRenderMode:TUI\_RENDER\_MODE\_FILL\_EDGE]; [LiveStrategyManager setRenderMode:V2TXLiveFillModeFill];

# 4.Data Management

#### 4.1. Data Model

The original data model of TUIShortVideoView includes:

Parameter name

Implication



| TUIPlayerDataModel  | Basic data types                                   |
|---------------------|--|
| TUIPlayerVideoModel | Video data type, inherited from TUIPlayerDataModel |
| TUIPlayerLiveModel  | Vive data type, inherited from TUIPlayerDataModel  |

## TUIPlayerDataModel

| Parameter name        | Implication                                  |
|-----------------------|--|
| modelType             | Model Type                                   |
| extInfo               | Business data                                |
| onExtInfoChangedBlock | ExtInfo The block where the data has changed |
| extInfoChangeNotify   | Notify extInfo that data has changed         |
| asVodModel            | Force conversion to TUIPlayerVideoModel type |
| asLiveModel           | Force conversion to TUIPlayerLiveModel type  |

## TUIPlayerVideoModel

| Parameter name  | Implication   |
|-----------------|---|
| videoUrl        | Video URL   |
| coverPictureUrl | Cover picture url   |
| duration        | duration  |
| appld           | appid   |
| fileId          | fileId  |
| pSign           | Signature String  |
| subtitles       | Subtitle information  |
| config          | Separate configuration of video, see TUIPlayerVideoConfig for details |

## TUIPlayerLiveModel

| Parameter name | Implication |
|----------------|-------------|
| liveUrl        | Live URL    |
|                |             |

Stencent Cloud

coverPictureUrl

cover picture url

### 4.2. Model Construction

Build a set of on-demand data models



```
TUIPlayerVideoModel *model = [[TUIPlayerVideoModel alloc] init];
model.videoUrl = @"xxxx";
model.coverPictureUrl = @"xxxx";
model.duration = @"xxxx";
```



```
model.appId = @"xxxx";
model.fileId = @"xxxx";
model.pSign = @"xxxx";
NSDictionary *extr = @{
    @"name":@"@Mars",
    @"titile":@"This is a vod broadcast interface",
    @"des":@"This is a vod broadcast interface"
};
model.extInfo = extr;
[modelArray addObject:model];
```

### Build a set of live data models



```
TUIPlayerLiveModel *model = [[TUIPlayerLiveModel alloc] init];
model.liveUrl = @"xxxx";
model.coverPictureUrl = @"xxxx";
NSDictionary *extr = @{
    @"name":@"@Mars",
    @"liveTitile":@"This is a live broadcast interface",
    @"liveDes":@"This is a live broadcast interface"
};
model.extInfo = extr;
```



Build a set of other types of data models



```
/// 1 Carousel
TUIPlayerDataModel *model = [[TUIPlayerDataModel alloc] init];
NSDictionary *extr = @{
    @"images":@"xxxx",
    @"url":@"https://cloud.tencent.com",
    @"titile":@"This is a picture carousel display interface",
    @"des":@"This is a picture carousel display interface",
    @"name":@"@Mars",
    @"type":@"imageCycle"
```

# S Tencent Cloud

```
};
model.extInfo = extr;
[modelArray insertObject:model atIndex:1];
/// 2 Graphic Ads
TUIPlayerDataModel *model1 = [[TUIPlayerDataModel alloc] init];
NSDictionary *extr1 = @{
    @"adUrl":@"https://cloud.tencent.com",
    @"adUrl":@"https://cloud.tencent.com/document/product",
    @"adUrl":@"https://cloud.tencent.com/document/product",
    @"adTitile":@"This is a web display interface",
    @"adDes":@"This is a web display interface",
    @"name":@"@Mars",
    @"type":@"ad"
};
model1.extInfo = extr1;
[modelArray insertObject:model1 atIndex:1];
```

#### Attention :

TUIPlayerDataModel is a large class that applies to all non-on-demand and live data types.

Users can use extInfo to perform more detailed classification. For example, in the figure above, two types of data,

"carousel" and "graphic advertisement", are constructed through TUIPlayerDataModel, which can be classified by extInfo/type.

extInfo is a flexible word, and users can design the data structure they need at will.

#### 4.3. Dynamic adjustment of data

TUIShortVideoView provides a data management class TUIShortVideoDataManager for external data operations. Its main function is to perform basic operations such as adding, deleting, modifying, and checking the data in the current player list. See the following demonstration code:





///1、Delete the data and view at index 1
[[self.videoView getDataManager] removeData:1];
///2、Add a set of data to index 9
TUIPlayerVideoModel \*model = [[TUIPlayerVideoModel alloc] init];
model.viewType = TUI\_ITEM\_VIEW\_TYPE\_CUSTOM;
[[self.videoView getDataManager] addData:model index:9];

A more detailed interface description is as follows:

Parameter name Implication

| removeData                  | Remove data by index                                       |
|-----------------------------|--|
| removeRangeData             | Remove data by range                                       |
| removeDataByIndex           | Remove data by index array                                 |
| addData:index               | Add data by index  |
| addRangeData:startIndex     | Add data from a certain index according to the model array |
| replaceData:index           | Replace data by index                                      |
| replaceRangeData:startIndex | Replace data from a certain index in the model array       |
| getDataByPageIndex          | Read data at a certain index                               |
| getCurrentDataCount         | Get the total number of data in the current playlist       |
| getCurrentIndex             | Get the data index of the current playback interface       |
| getCurrentModel             | Get the data model of the current playback interface       |
|                             |  |

#### Explanation :

Tencent Cloud

The UI interface is automatically refreshed after the DataManager interface is called.

If the current playback interface is not operated, there will be no refresh.

Operating the current interface will refresh the current interface.

If the current playback interface is deleted, the next one will be played automatically. If there is no data for the next one (it has reached the end), the previous one will be played.

## 5. Custom UI Layers

## 5.1. Hierarchy

The hierarchical structure of TUIPlayerShortVideo is as follows:





It is divided into a display layer and a control layer, and the two are combined in a stacked manner.

The display layer is responsible for content display, on-demand, live broadcast, advertising promotion pages, etc. This layer is managed internally by the SDK.

The control layer is responsible for interaction, likes, comments, etc. This layer is left to the user to achieve a high degree of customization.

## 5.2. TUIPlayerShortVideoUIManager

You can use the TUIPlayerShortVideoUIManager interface and the protocol control layer interface to implement your custom UI. See the following code:





```
TUIPlayerShortVideoUIManager *uiManager = [[TUIPlayerShortVideoUIManager alloc] ini
[uiManager setControlViewClass: TUIPSControlView.class];
[uiManager setLoadingView:[[TUIPSLoadingView alloc] init]];
[uiManager setBackgroundView:[UIView new]];
_videoView = [[TUIShortVideoView alloc] initWithUIManager:uiManager];
```

The above code customizes the video control layer (progress bar, time, etc.) named TUIPSControlView and the loading control named TUIPSLoadingView through TUIPlayerShortVideoUIManager.

The interface of TUIPlayerShortVideoUIManager is shown in the following table:

Parameter name

Implication



| setLoadingView                   | Setting up the loading graph  |
|----------------------------------|---|
| setBackgroundView                | Set the background image  |
| setErrorView                     | Setting the error interface   |
| setControlViewClass              | Set the video control layer * @param ViewClass control layer class,<br>ViewClass is the video control View you have encapsulated, including<br>controls such as progress bar, time label, etc. * It will be covered on the<br>video window as a whole, and its size is consistent with the video window |
| setControlViewClass<br>:viewType | Set up different types of video control layers  |
| getLoadingView                   | Get the loading image View instance   |
| getBackgroundView                | Get the background image View instance  |
| getErrorView                     | Get the error interface View instance   |
| getControlViewClass              | Get the video control interface View class  |
| getControlViewClassWithViewType  | Get different types of video control interface classes  |

The playback layer currently supports two types:

TUI\_ITEM\_VIEW\_TYPE\_VOD ///video

TUI\_ITEM\_VIEW\_TYPE\_LIVE ///live

TUI\_ITEM\_VIEW\_TYPE\_CUSTOM /// Custom type (such as advertising page)

The corresponding control layer protocol also supports two types:

TUIPlayerShortVideoControl ///Video Control Layer Protocol

TUIPlayerShortVideoLiveControl ///Live broadcast control layer protocol

TUIPlayerShortVideoCustomControl ///Custom control layer protocol

#### explanation:

All non-VOD and live interfaces are presented with TUI\_ITEM\_VIEW\_TYPE\_VOD and

TUIPlayerShortVideoCustomControl. Custom is a large category, and the specific subdivisions need to be defined by the user on this whiteboard. As mentioned in the TUIPlayerDataModel model construction above, more fine-grained divisions can be made through extInfo/type or other fields under extInfo.

The specific interface description of TUIPlayerShortVideoControl protocol is as follows:

| Parameter name | Implication  |
|----------------|--|
| delegate       | A reverse proxy for interaction between the control layer and the playback layer |
|                |  |

| model               | The currently playing video model           |
|---------------------|---|
| currentPlayerStatus | The current player's playback status        |
| showCenterView      | Display center view                         |
| hideCenterView      | Hide center view                            |
| showLoadingView     | Display loading graph                       |
| hiddenLoadingView   | Hide loading image                          |
| setDurationTime     | Total video time                            |
| setCurrentTime      | Current playback time                       |
| setProgress         | Progress bar progress                       |
| showSlider          | Show progress bar                           |
| hideSlider          | Hide progress bar                           |
| reloadControlData   | Triggering a view refresh                   |
| getPlayer           | Get the player object                       |
| onPlayEvent         | Get player events                           |
| getVideoLayerRect   | Get the changes of the video rendering area |
| getVideoWidget      | Get the video rendering layer object        |

The specific interface description of TUIPlayerShortVideoLiveControl protocol is as follows:

| Parameter name    | Implication  |
|-------------------|--|
| delegate          | A reverse proxy for interaction between the control layer and the playback layer |
| model             | Current playback data model  |
| reloadControlData | Triggering a view refresh  |
| getPlayer         | Get the player object  |
| getVideoLayerRect | Get the video layer area   |
| getVideoWidget    | Get the video rendering layer  |

| Parameter name    | Implication  |
|-------------------|--|
| delegate          | A reverse proxy for interaction between the control layer and the playback layer |
| model             | Current playback data model  |
| reloadControlData | Triggering a view refresh  |

The specific interface description of TUIPlayerShortVideoCustomControl protocol is as follows:

#### Attention:

The custom control layer View passed in needs to comply with the relevant protocols, otherwise the compilation will fail.

#### 5.3. Example

| Type video live | Custom (slides)   |
|-----------------|---|
| example         | ##! * * *<br>brt Video<br>true the live room<br>不 远 也 不 着<br>terface(FLV) |

#### 5.3.1. Define UI styles under different styles

vod style (TUIPlayerShortVideoControl)



```
@interface TUIPSControlView : UIView<TUIPlayerShortVideoControl>
@property (nonatomic, strong) TUIPlayerVideoModel *videoModel
@end
@implementation TUIPSControlView
-(instancetype)initWithFrame:(CGRect)frame {
    if ([super initWithFrame:frame]){
        /// UI layout code
    }
    return self;
```

}

```
-(void)setModel:(TUIPlayerVideoModel *)model {
    __model = model;
    /// data
}
@end
```

## Live broadcast style (TUIPlayerShortVideoLiveControl)



@interface TUIPSControlLiveView : UIView<TUIPlayerShortVideoLiveControl>

```
@property (nonatomic, strong) TUIPlayerLiveModel *videoModel
@end
@implementation TUIPSControlLiveView
-(instancetype)initWithFrame:(CGRect)frame {
    if ([super initWithFrame:frame]) {
        /// UI layout code
    }
    return self;
}
-(void)setModel:(TUIPlayerLiveModel *)model {
    _model = model;
    /// data
}
@end
```

Other styles such as carousel & graphic ads (TUIPlayerShortVideoCustomControl)





```
@interface TUIPSControlCustomView : UIView<TUIPlayerShortVideoCustomControl>
@property (nonatomic, strong) TUIPlayerDataModel *videoModel
@end
@implementation TUIPSControlCustomView
-(instancetype)initWithFrame:(CGRect)frame {
    if ([super initWithFrame:frame]){
        /// UI layout code
    }
    return self;
}
```

```
-(void)setModel:(TUIPlayerDataModel *)model {
    model = model;
    /// data
   NSDictionary *dic = model.extInfo;
   NSString *adTitile = [dic objectForKey:@"adTitile"];
    NSString *adDes = [dic objectForKey:@"adDes"];
    NSString *adUrl = [dic objectForKey:@"adUrl"];
    NSString *name = [dic objectForKey:@"name"];
    NSString *type = [dic objectForKey:@"type"];
    if ([type isEqualToString:@"ad"]) { ///Graphic Ads
        self.webView.hidden = NO;
        self.cycleScrollView.hidden = YES;
        self.desLabel.textColor = [UIColor blackColor];
        self.nameLabel.textColor = [UIColor blackColor];
        [self.webView loadRequest: [NSURLRequest requestWithURL: [NSURL URLWithString
    } else if ([type isEqualToString:@"imageCycle"]) { ///Carousel
        self.webView.hidden = YES;
        self.cycleScrollView.hidden = NO;
        self.desLabel.textColor = [UIColor whiteColor];
        self.nameLabel.textColor = [UIColor whiteColor];
        NSString *imagesStr = [dic objectForKey:@"images"];
        NSArray *imagesArray = [imagesStr componentsSeparatedByString:@"<:>"];
        self.cycleScrollView.imageURLStringsGroup = imagesArray;
    }
}
0end
```

#### 5.3.2. Register the created style through TUIPlayerShortVideoUIManager





TUIPlayerShortVideoUIManager \*uiManager = [[TUIPlayerShortVideoUIManager alloc] ini [uiManager setControlViewClass: TUIPSControlView.class viewType:TUI\_ITEM\_VIEW\_TYPE\_ [uiManager setControlViewClass: TUIPSControlLiveView.class viewType:TUI\_ITEM\_VIEW\_T [uiManager setControlViewClass: TUIPSControlCustomView.class viewType:TUI\_ITEM\_VIEW]

5.3.2. Initialize TUIShortVideoView through TUIPlayerShortVideoUIManager





\_videoView = [[TUIShortVideoView alloc] initWithUIManager:uiManager];

5.3.3. Use setShortVideoStrategyModel and setShortVideoLiveStrategyModel to set the strategies for video on demand and live streaming.





```
// Set your playback strategy
TUIPlayerVodStrategyModel *model = [[TUIPlayerVodStrategyModel alloc] init];
model.mPreloadConcurrentCount = 1;
model.preDownloadSize = 1;
model.enableAutoBitrate = NO;
[_videoView setShortVideoStrategyModel:model];
// live strategy
TUIPlayerLiveStrateyModel *liveStrateyModel = [[TUIPlayerLiveStrateyModel alloc] i
[_videoView setShortVideoLiveStrategyModel:liveStrateyModel];
```

#### 5.3.4. The relationship between UI and data

TUIPSControlView & TUIPSControlLiveView & TUIPSControlCustomView can be understood as preset templates. When TUIShortVideoView is initialized, various types of templates have been preset. When the corresponding data type is slipped, the current template will be displayed.

| Туре                                       | Data model          | UI Templates           |
|--|---------------------|------------------------|
| Vod  | TUIPlayerVideoModel | TUIPSControlView       |
| Live                                       | TUIPlayerLiveModel  | TUIPSControlLiveView   |
| Custom types (carousel, graphic ads, etc.) | TUIPlayerDataModel  | TUIPSControlCustomView |

#### UI templates are pre-set

The UI style is displayed according to the driver of the corresponding data type.

#### 5.3.4. Interaction between UI template and TUIShortVideoView

The custom UI template interacts with TUIShortVideoView through the corresponding protocol. TUIShortVideoView passes messages to TUIPSControlView & TUIPSControlLiveView & TUIPSControlCustomView through the protocol methods of TUIPlayerShortVideoControl &TUIPlayerShortVideoLiveControl &TUIPlayerShortVideoCustomControl. As VOD:





```
-(void)setModel:(TUIPlayerVideoModel *)model {
    if ([_model observationInfo]) {
        [_model removeObserver:self forKeyPath:@"preloadState"];
    }
    _model = model;
    [model addObserver:self forKeyPath:@"preloadState" options:NSKeyValueObservingO
```



```
NSDictionary *dic = model.extInfo;
NSString *iconUrl = [dic objectForKey:@"iconUrl"];
NSString *advertise = [dic objectForKey:@"advertise"];
NSString *name = [dic objectForKey:@"name"];
NSString *title = [dic objectForKey:@"title"];
NSString *topic = [dic objectForKey:@"topic"];
self.iconImageView.image = [UIImage imageNamed:iconUrl];
[self.adButton setTitle:advertise forState:UIControlStateNormal];
self.nameLabel.text= name;
self.themeLabel.text = topic;
self.desLabel.text = title;
[self updatePreloadState];
[self updateLickCount];
model.onExtInfoChangedBlock = ^(id _Nonnull extInfo) {
    [self updateLickCount];
};
```

Live :

}



```
-(void)setModel:(TUIPlayerLiveModel *)model {
    _model = model;
    NSDictionary *dic = model.extInfo;
    NSString *adTitile = [dic objectForKey:@"liveTitile"];
    NSString *adDes = [dic objectForKey:@"liveDes"];
    NSString *name = [dic objectForKey:@"name"];
    self.nameLabel.text = name;
    self.desLabel.text = adTitile;
```


Video on Demand



Custom (carousel, pictures and text, etc.)



```
-(void)setModel:(TUIPlayerDataModel *)model {
    _model = model;
    NSDictionary *dic = model.extInfo;
    NSString *adTitile = [dic objectForKey:@"adTitile"];
    NSString *adDes = [dic objectForKey:@"adDes"];
    NSString *adUrl = [dic objectForKey:@"adUrl"];
```

```
NSString *name = [dic objectForKey:@"name"];
    NSString *type = [dic objectForKey:@"type"];
    self.desLabel.text = adTitile;
    self.nameLabel.text = name;
    if ([type isEqualToString:@"web"]) {
        self.webView.hidden = NO;
        self.cycleScrollView.hidden = YES;
        self.desLabel.textColor = [UIColor blackColor];
        self.nameLabel.textColor = [UIColor blackColor];
        [self.webView loadRequest: [NSURLRequest requestWithURL: [NSURL URLWithString
    } else if ([type isEqualToString:@"imageCycle"]) {
        self.webView.hidden = YES;
        self.cycleScrollView.hidden = NO;
        self.desLabel.textColor = [UIColor whiteColor];
        self.nameLabel.textColor = [UIColor whiteColor];
        NSString *imagesStr = [dic objectForKey:@"images"];
        NSArray *imagesArray = [imagesStr componentsSeparatedByString:@"<:>"];
        self.cycleScrollView.imageURLStringsGroup = imagesArray;
    }
}
```

TUIPSControlView & TUIPSControlLiveView & TUIPSControlCustomView pass messages to TUIShortVideoView through the delegate of TUIPlayerShortVideoControl &TUIPlayerShortVideoLiveControl &TUIPlayerShortVideoCustomControl. As VOD :



```
@protocol TUIPlayerShortVideoControlDelegate <NSObject>
/**
 * Pause
 */
- (void)pause;
/**
 * Resume
 */
- (void)resume;
```



```
/**
* Sliding scroll bar processing
 * @param time Slide distance
 */
- (void) seekToTime: (float) time;
/**
* Is it playing
*/
- (BOOL) is Playing;
/**
* Reset video playback container
* Used for scenarios where the video playback container needs to be reset after be
 */
- (void) resetVideoWeigetContainer;
Coptional
/**
* Custom callback events
*/
- (void)customCallbackEvent:(id)info;
0end
////transfer
if (self.delegate && [self.delegate respondsToSelector:@selector(pause)]) {
    [self.delegate pause];
}
```

Live



```
@protocol TUIPlayerShortVideoLiveControlDelegate <NSObject>
/**
 * pause
 */
- (void)pause;
/**
 * resume
 */
```



```
- (void) resume;
/**
* Reset video playback container
* Used for scenarios where the video playback container needs to be reset after be
*/
- (void) resetVideoWeigetContainer;
@optional
/**
* Custom callback events
*/
- (void)customCallbackEvent:(id)info;
0end
////transfer
if (self.delegate && [self.delegate respondsToSelector:@selector(pause)]) {
    [self.delegate pause];
}
```

Custom (carousel, pictures and text, etc.)



```
@protocol TUIPlayerShortVideoCustomControlDelegate <NSObject>
@optional
/**
 * Custom callback events
 */
- (void)customCallbackEvent:(id)info;
@end
/////transfer
```

```
if (self.delegate && [self.delegate respondsToSelector:@selector(customCallbackEven
    [self.delegate customCallbackEvent:@"test"];
}
```

### explanation:

See Demo for the complete example.

## **Advanced Features**

### Business notification message to the page layer

TUI provides a message interface for users to notify the current layer of data in real time. After obtaining the video object through the data operation object, the notification can be made. The example is as follows:





```
/// Get Data Manager
TUIShortVideoDataManager *dataManager = [self.videoView getDataManager];
///Get the data model
TUIPlayerDataModel *model = [dataManager getDataByPageIndex:1];
///Modify the data model
model.extInfo = @{@"key":@"value"}
///Notify data model changes
[model extInfoChangeNotify];
```

The notification is then received in the onExtInfoChanged callback of the UI control layer, so that the UI of the current page can be modified. The example is as follows:



```
model.onExtInfoChangedBlock = ^(id _Nonnull extInfo) {
    [self updateLickCount];
};
```

## Attention:

This function is only valid for the extInfo field.

## Volume balance on demand

The player supports automatic volume adjustment when playing audio, so that the volume of all audio is consistent. This can avoid the problem of some audio being too loud or too quiet, and provide a better listening experience. By setting the volume balance, the loudness range is:  $-70 \sim 0$  (LUFS), and custom values are also supported.

### Attention :

Supported by Player Premium 11.7.



/// Volume balance. Loudness range:  $-70 \sim 0$  (LUFS). This configuration requires LiteA<sup>v</sup>

STencent Cloud

/// The following constants are for reference /// Off: AUDIO\_NORMALIZATION\_OFF (TXVodPlayConfig.h) /// On (standard loudness): AUDIO\_NORMALIZATION\_STANDARD (TXVodPlayConfig.h) /// On (low loudness): AUDIO\_NORMALIZATION\_LOW (TXVodPlayConfig.h) /// On (high loudness): AUDIO\_NORMALIZATION\_HIGH (TXVodPlayConfig.h) /// The default value is AUDIO\_NORMALIZATION\_OFF.

TUIPlayerVodStrategyModel \*model = [[TUIPlayerVodStrategyModel alloc] init]; model.audioNormalization = AUDIO\_NORMALIZATION\_STANDARD;

[\_videoView setShortVideoStrategyModel:model];

# Android

Last updated : 2024-06-17 17:49:53

## **Component Introduction**

The TUIPlayerShortVideo component is an excellent-performance short video component launched by Tencent Cloud, which supports ultra-fast first frame display and smooth sliding, providing a high-quality playback experience. Instant First Frame: The time to the first frame is one of the core indicators for short video applications, directly affecting the user's viewing experience. The short video component achieves ultra-fast first frame and smooth sliding playback through technologies such as pre-playing, pre-downloading, player reuse, and precise traffic control, thereby enhancing user playback volume and dwell time.

Excellent Performance: Through player reuse and optimization of loading strategies, while ensuring excellent smoothness, memory and CPU consumption are always kept at a low level.

Rapid Integration: The component encapsulates complex playback operations, provides a default playback UI, and supports both FileId and Url playback, allowing for low-cost and rapid integration into your project.

## Effect Comparison

The following video demonstrates the comparative differences in short video usage under the same environment, before and after optimization.

Before optimization, there is a noticeable lag when starting the video playback.

After optimization, a seamless startup experience can be achieved, with the average startup duration after optimization reaching 10 milliseconds to 30 milliseconds.

| Unoptimized Short Video | Optimized Short Video |
|-------------------------|-----------------------|
|                         |                       |
|                         |                       |
|                         |                       |
|                         |                       |
|                         |                       |

## TUIPlayerKit Download



The TUIPlayerKit SDK and Demo can be downloaded by clicking here.

## Integrate the TUIPlayerShortVideo Component

## **Environment Preparation**

Minimum Android system version required: Android SDK >= 19

Add the dependencies required for short videos:



// If you are using the professional version of the SDK,



```
// use: api 'com.tencent.liteav:LiteAVSDK_Professional:latest.release'
api 'com.tencent.liteav:LiteAVSDK_Player:latest.release'
implementation (name:'tuiplayercore-release_x.x.x', ext:'aar')
implementation (name:'tuiplayershortvideo-release_x.x.x', ext:'aar')
implementation 'androidx.appcompat:appcompat:1.0.0'
implementation 'androidx.viewpager2:viewpager2:1.0.0'
```

#### Note :

Where the x.x.x in tuiplayercore-release and tuiplayershortvideo-release represents the version number, note that the version numbers of the two aar files must match. Permissions Required by the SDK:





```
<uses-permission android:name="android.permission.INTERNET" />
```

Set ProGuard rules:

In the proguard-rules.pro file, add the relevant classes to the list of classes not to be obfuscated:



```
-keep class com.tencent.** { *; }
```

## Apply for the Advanced Player License

To use the TUIPlayer Kit component, you need to use the Advanced Mobile Player License. You can refer to the advanced version of the mobile player License to obtain it. If you have already obtained the corresponding License, you can go to Tencent Cloud Visual Cube Console > License Management > Mobile License to get the corresponding LicenseURL and LicenseKey. If you have not applied for the Advanced Mobile Player License, issues such as video playback failure and black screen may occur.

## Set Up License

The short video component needs to have a License set up before it can be used.



TUIPlayerConfig config = new TUIPlayerConfig.Builder()

```
.enableLog(true)
.licenseKey("Your license key")
.licenseUrl("Your license url")
.build();
TUIPlayerCore.init(context, config);
```

## In your layout file

add the short video UI component.



<com.tencent.qcloud.tuiplayer.shortvideo.ui.view.TUIShortVideoView

## 🕗 Tencent Cloud

```
android:id="@+id/my_video_view"
android:layout_height="match_parent"
android:layout_width="match_parent"/>
```

## Set up the lifecycle

After setting up the lifecycle, the component will automatically pause, resume playback, and release itself based on the current Activity lifecycle. For example, when the app goes to the background, the short video will pause playback automatically. When you return to the app, the video will continue playing. You can also choose not to set this lifecycle and control the short video according to your business needs.





```
mSuperShortVideoView.setActivityLifecycle(getLifecycle());
```

## **Configure listening**

The TUIShortVideoView listener offers multiple callbacks, as illustrated in the code below:



```
mSuperShortVideoView.setListener(new TUIShortVideoListener() {
  @Override
  public void onPageChanged(int index, TUIVideoSource videoSource) {
    if (index >= mSuperShortVideoView.getCurrentDataCount() - 1) {
      // append next page data
```

```
mSuperShortVideoView.appendModels(data);
    }
 }
QOverride
public void onCreateVodLayer(TUIVodLayerManager layerManger, int viewType) {
    // add your vod layer to here
     layerManger.addLayer(new TUICoverLayer());
}
@Override
public void onCreateLiveLayer(TUILiveLayerManager layerManager, int viewType) {
    // add your live layer to here
}
Override
public void onCreateCustomLayer(TUICustomLayerManager layerManager, int viewType)
    // add your custom layer to here
 }
@Override
public void onNetStatus(TUIVideoSource model, Bundle bundle) {
}
});
```

Whenever the page position changes, the onPageChanged method will be called back, where you can implement capabilities similar to paginated loading.

When the list creates its layout, it will call back the onCreateVodLayer, onCreateLiveLayer, or onCreateCustomLayer methods based on the type of data you add. Specifically, if the data is of type TUIVideoSource when setting models, onCreateVodLayer will be called back; if it's of type TUILiveSource, onCreateLiveLayer will be called back; and if it's a custom data type implemented by inheriting from TUIPlaySource, onCreateCustomLayer will be called back.

Here, the extViewType from TUIPlayerSource will also be called back, which can be used for the business to distinguish different layer groups based on the viewType. The creation of layers will be discussed in the section on custom layers.

```
After the video starts playing, the network status of the currently playing video will be called back through onNetStatus. For details on the callback, please refer to here.
```

## Populate data

Using setModels can set the data and clear the original data. The video will also restart playback from the first video of the newly set data source. Using appendModels can append data, which is used for pagination



operations. After filling the data, it will automatically start playing from the first video. If automatic playback is not needed, you can call setAutoPlay on the first video and set it to false .



```
// vod
TUIVideoSource videoSource = new TUIVideoSource();
videoSource.setCoverPictureUrl(model.placeholderImage);
// Fill in the fileId. Only one of fileId and url needs to be filled in.
videoSource.setAppId(model.appid);
videoSource.setPSign(model.pSign);
videoSource.setFileId(model.fileid);
// Fill in the fileId url
```

```
videoSource.setVideoURL(model.videoURL);
shortVideoData.add(videoSource);
// live
TUILiveSource liveSource = new TUILiveSource();
// live url
liveSource.setUrl(liveUrl);
liveSource.setCoverPictureUrl(coverUrl);
shortVideoData.add(liveSource);
// Custom data. DemoImgSource inherits from TUIPlaySource, providing custom data.
// Here, different data can be customized according to business requirements.
DemoImgSource imgSource = new DemoImgSource("imgUrl");
shortVideoData.add(imgSource);
// set data
mSuperShortVideoView.setModels(shortVideoData);
// For pagination operations, you can choose to append data.
```

```
mSuperShortVideoView.appendModels(shortVideoData);
```

## Custom Layer

## I. Introduction

## 1. Layer Categories

If it is necessary to customize the UI for TUI short videos, it is required to use the layer capabilities of the short videos. Android TUI short videos adopt a layer management approach, providing each short video page with the ability to customize UI. Through the layer manager, each page can be object-managed, better handling the interaction between video UI and player and video components.

Currently, there are three types of pages: on-demand, live broadcast, and custom pages. The corresponding layers for these three pages are TUIVodLayer, TUILiveLayer, and

**TUICustomLayer**, respectively. Different layers can be inherited according to requirements. Different layer base classes provide interfaces and callbacks that fit their scenarios.

The corresponding relationship is shown in the figure below:



The display and hiding of layers will directly add and remove Views, without causing excessive interface rendering. Layers determine the display order of the interface based on the order they are added, with those added first appearing on the top layer and prioritized for display, while those added later are on the bottom layer and will be covered by those added before.

Due to the page reuse mechanism in the short video list, when there is business data-related UI display in the layer, it is necessary to reset the interface UI or set new values when binding data in onBindData.

## 2. Layer Lifecycle

The lifecycle of the three types of Layers is shown in the figure below:





The meanings of the lifecycle are as follows:

| LifeCycle Name     | Description  |
|--------------------|--|
| onBindData         | Indicates that the current Layer has been bound to data. In this lifecycle, some static UI data initialization work can be done.   |
| onControllerBind   | This page is already the one currently displayed in the<br>short video list. Except for the custom page<br>TUICustomLayer, calling getPlayer()<br>and getController() at this time will no<br>longer return null, allowing operations on the player<br>and page container. |
| onControllerUnBind | The page has been swiped away, and after this lifecycle, it will no longer be possible to obtain the player and videoView. This lifecycle can be used to   |



|                       | perform resource recycling and interface resetting tasks.   |
|-----------------------|---|
| onViewRecycled        | The page has been recycled and will be used with<br>other data and players. It is recommended to reset all<br>data on the interface and recycle related resources in<br>this lifecycle. |
| onShortVideoDestroyed | The TUI component has been destroyed, meaning the TUIShortVideoView's release method has been called.   |

## II. Creating a Custom Layer

#### 1. Create custom layer layouts

To create a custom layer, taking the on-demand layer as an example, you need to inherit from TUIVodLayer and then implement the layer you need.

Taking the video details layer as an example, you first need to implement the createView and tag methods. createView is the method for creating the layer view, and tag is a string label used by the business side to distinguish layers. createView will be called when the layer is added to the LayerManager.



```
@Override
public View createView(ViewGroup parent) {
    LayoutInflater inflater = LayoutInflater.from(parent.getContext());
    View view = inflater.inflate(R.layout.player_video_info_layer, parent, false);
    mSeekBar = view.findViewById(R.id.vsb_tui_video_progress);
    mTvProgress = view.findViewById(R.id.tv_tui_progress_time);
    mIvPause = view.findViewById(R.id.iv_tui_pause);
    mSeekBar.setListener(this);
    return view;
}
```

## 🔗 Tencent Cloud

```
@Override
public String tag() {
    return "TUIVideoInfoLayer";
}
```

In createView, a View is created and returned. Here, you can use LayoutInflater to load a layout from XML, or you can create the layout directly using code.

## 2. Display Layout

After the View is created, it needs to be displayed at the appropriate time. TUIBaseLayer provides a rich set of event callbacks. For detailed information, see Layer Callbacks.

The video details layer can display the layout once it has obtained the data. Therefore, display the layer in onBindData.





```
@Override
public void onBindData(TUIVideoSource videoSource) {
    show();
}
```

#### 3. Operating your own components

You can also operate on components you have created in other events. Taking the on-demand layer as an example, you need to declare your component as a member variable first, assign a value in onCreateView, and then handle



player events such as the display and hiding of the pause button, as well as the callback for playback progress. See the following code.



```
@Override
public void onPlayBegin() {
    super.onPlayBegin();
    if (null != mIvPause) {
        mIvPause.setVisibility(View.GONE);
    }
}
```

```
@Override
public void onPlayPause() {
    super.onPlayPause();
    if (null != mIvPause) {
       mIvPause.setVisibility(View.VISIBLE);
    }
}
@Override
public void onPlayProgress(long current, long duration, long playable) {
   videoDuration = duration;
    if (null != mSeekBar) {
        // ensure a refresh at every percentage point
        int progressInt = (int) (((1.0F * current) / duration) * 100);
        if(lastProgressInt != progressInt) {
            setProgress(progressInt / 100F);
            lastProgressInt = progressInt;
        }
    }
}
```

### 4. Controlling the player

In addition to receiving events from the player, you can also control the player. For example, call the player to perform a seek operation.





```
@Override
public void onDragDone(VideoSeekBar seekBar) {
   TUIPlayerController controller = getPlayerController();
   if (null != controller && videoDuration > 0) {
      controller.seekTo((int) ((videoDuration * seekBar.getBarProgress()) / 1000)
    }
   if (null != mTvProgress) {
      mTvProgress.setVisibility(View.GONE);
    }
}
```

Currently, in TUIBaseLayer, you can obtain the VideoView object of the current page through getVideoView, obtain the playback controller of the current video through getPlayerController (only available when the current page is the video currently playing in the short video list), and get the current player object through getPlayer. Since the player and video view may be released or reused during scrolling, these three objects might be null when obtained, so it is necessary to check for null values.

### 5. Release when the layer is recycled

When the layer is recycled, some release operations need to be performed. This prevents external objects from holding onto the layer, causing memory leaks.



```
@Override
public void onViewRecycled(TUIBaseVideoView videoView) {
    // release your resource
}
```

#### 6. Listening for whether the current layer is scrolled to

If you need to listen for whether the current page is the currently playing video, you can listen to the controller. When onControllerBind is triggered, the layer is bound by the controller, indicating that the page of the layer is about to be displayed and start playing. When onControllerUnBind is triggered, the controller is unbound, indicating that the page has been swiped away.





```
@Override
public void onControllerUnBind(TUIPlayerController controller) {
    super.onControllerUnBind(controller);
    show();
}
```

The above code is executed after the page is swiped away and the controller is unbound. To prevent the screen from going black when the page is swiped back later, it triggers the display of the cover image.

#### 7. Vod playback obtains video information through the onRecFileVideoInfo callback.

When a fileId video source is set, the layer will callback the video information through onRecFileVideoInfo. See the following code:




```
Glide.with(videoView).load(coverUrl)
        .centerCrop()
        .into(imageView);
        coverUrlFromServer = coverUrl;
     }
}
}
```

This method will be called back when playing only using fileID. It will return information such as the video URL link, cover image, duration, and sprite image.

It is recommended to pass the short video component for playback through the URL as much as possible, and preassign the cover image URL in advance, which can improve the loading performance of short videos.

#### 8. Determine if the first frame of the video has arrived through the onRcvFirstIframe method

Usage example:



```
@Override
public void onRcvFirstIframe() {
    hidden();
}
```

For example, in scenarios such as cover images, you need to hide the cover image after receiving the first frame event.

# **III. Managing Layers**

After integrating the short video component TUIShortVideoView, setting up a listener with TUIShortVideoView will callback the item creation method onCreateItemLayer at the appropriate time, allowing you to add or manage custom layers.



mSuperShortVideoView.setListener(new TUIShortVideoListener() {

// .....

@Override

public void onCreateVodLayer(TUIVodLayerManager layerManger, int viewType) {
 layerManger.addLayer(new TUIVideoInfoLayer(mShortVideoView, ShortVideoFragmen

```
layerManger.addLayer(new TUICoverLayer());
      layerManger.addLayer(new TUILoadingLayer());
      layerManger.addLayer(new TUIErrorLayer());
  }
 QOverride
 public void onCreateLiveLayer(TUILiveLayerManager layerManager, int viewType) {
      layerManager.addLayer(new TUILiveEntranceLayer(mShortVideoView, ShortVideoFra
      layerManager.addLayer(new TUILiveLoadingLayer());
      layerManager.addLayer(new TUILiveErrorLayer());
  }
 Override
 public void onCreateCustomLayer(TUICustomLayerManager layerManager, int viewType)
      if (viewType == SVDemoConstants.CustomSourceType.SINGLE_IMG_TYPE) {
          layerManager.addLayer(new PicDisplayLayer());
      }
  }
});
```

onCreateItemLayer has two parameters: layerManager, which is the layer manager that can add, remove, and query layers. The method of adding is shown in the figure above. viewType is the video type of the current page. If you have customized extViewType in TUIPlayerSource, the viewType here will be what you defined. If not defined, it will return ITEM\_TYPE\_VOD, ITEM\_TYPE\_LIVE, or ITEM\_TYPE\_CUSTOM according to the page type.

If no layer is needed, you can remove the layer. After removal, the unbindLayerManager method will be called back in the layer.





```
layerManger.removeLayer(layer);
```

If you need to obtain the layer hierarchy for interactive operations on layers, you can get the layer hierarchy through the following methods:



layerManger.indexOfLayer(layer);

# IV. Creating an Image Display Page Using Custom Layers

## 1. Implement your own custom data

Taking displaying images as an example, create data with image links.





```
public class DemoImgSource extends TUIPlaySource {
    private String mImgUrl;
    public DemoImgSource(String imgUrl) {
        mImgUrl = imgUrl;
        // You can specify different viewType parameters to distinguish between typ
        setExtViewType(SVDemoConstants.CustomSourceType.SINGLE_IMG_TYPE);
    }
    public String getImgUrl() {
```

```
return mImgUrl;
}
public void setImgUrl(String imgUrl) {
   this.mImgUrl = imgUrl;
}
```

#### 2. Implement the UI of a custom page

After implementing the data, it is necessary to customize the UI of your own custom page. Taking displaying images as an example, you need to inherit from TUICustomLayer to implement the layer.





```
public class PicDisplayLayer extends TUICustomLayer {
    private ImageView mDisplayImgView;
    @Override
    public View createView(ViewGroup parent) {
        // Constructing a Page view
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());
        View view = inflater.inflate(R.layout.tuiplayer_img_display_layer, parent,
        mDisplayImgView = view.findViewById(R.id.iv_img_display);
        return view;
```

```
}
@Override
public void onBindData(TUIPlaySource videoSource) {
    super.onBindData(videoSource);
    // Data is bound to the page, allowing access to the corresponding data sou
    if (videoSource.getExtViewType() == SVDemoConstants.CustomSourceType.SINGLE
        DemoImgSource source = (DemoImgSource) videoSource;
        Glide.with(mDisplayImgView).load(source.getImgUrl())
            .into(mDisplayImgView);
    }
}
@Override
public String tag() {
    return "PicDisplayLayer";
    }
}
```

Add your own layer into the TUI short video callback





```
mSuperShortVideoView.setListener(new TUIShortVideoListener() {
```

// .....

```
@Override
public void onCreateCustomLayer(TUICustomLayerManager layerManager, int viewType)
    // custom layer
    if (viewType == SVDemoConstants.CustomSourceType.SINGLE_IMG_TYPE) {
        layerManager.addLayer(new PicDisplayLayer());
    }
}
```



Video on Demand

});

#### 3. Populate data into TUI short video



// Customize data, DemoImgSource inherits from TUIPlaySource, customize data, // different data can be customized here according to business needs DemoImgSource imgSource = new DemoImgSource("imgUrl"); shortVideoData.add(imgSource);

#### // fill data

mSuperShortVideoView.setModels(shortVideoData);

Subsequently, the custom page will be displayed at the corresponding page position in the short video list, according to the position you added in the list.

# TUI short video interface

# 1. Configure License

To use the TUI component, you need to configure the corresponding premium license. An example is as follows:



```
TUIPlayerConfig config = new TUIPlayerConfig.Builder()
        .enableLog(true)
        .licenseKey(LICENCE_KEY)
        .licenseUrl(LICENCE_URL)
        .build();
TUIPlayerCore.init(getApplicationContext(), config);
```

# 2. Set up lifecycle listening

Used for lifecycle control of TUIShortVideoView. Internally, it automatically pauses, plays, and destroys the list videos according to the lifeCycle status. This interface can be unset, and the business can take over the call.





mSuperShortVideoView.setActivityLifecycle(getLifecycle());

# 3. Set up short video listening

Used to listen to events from TUIShortVideoView, including the timing of loading paginated data, and the callback during page creation, where layers can be added.





```
mSuperShortVideoView.setListener(new TUIShortVideoListener() {
    @Override
    public void onCreateVodLayer(TUIVodLayerManager layerManger, int viewType) {
        layerManger.addLayer(new TUIVideoInfoLayer(mShortVideoView, ShortVideoFragm
        layerManger.addLayer(new TUICoverLayer());
        layerManger.addLayer(new TUILoadingLayer());
        layerManger.addLayer(new TUIErrorLayer());
    }
    @Override
    public void onCreateLiveLayer(TUILiveLayerManager layerManager, int viewType) {
}
```

```
layerManager.addLayer(new TUILiveEntranceLayer(mShortVideoView, ShortVideoF
        layerManager.addLayer(new TUILiveLoadingLayer());
        layerManager.addLayer(new TUILiveErrorLayer());
    }
    @Override
    public void onCreateCustomLayer(TUICustomLayerManager layerManager, int viewTyp
        if (viewType == SVDemoConstants.CustomSourceType.SINGLE_IMG_TYPE) {
            layerManager.addLayer(new PicDisplayLayer());
        }
    }
    QOverride
    public void onPageChanged(int index, TUIPlaySource videoSource) {
        if (index >= mShortVideoView.getCurrentDataCount() - 1) {
            mShortViewRefresh.setRefreshing(true);
            ShortVideoModel.getInstance().loadMore(false);
        }
    }
    @Override
    public void onNetStatus(TUIPlaySource model, Bundle bundle) {
    }
});
```

### 4. Set the video playback strategy

Set various strategies during the video playback process.

#### The TUIPlayerVodStrategy parameter.

It needs to be constructed using the Builder.

| Method                 | Description   |
|------------------------|---|
| setPreloadCount        | Set the maximum concurrent preload quantity, default is 3.                              |
| setPreDownloadSize     | Set the preload cache size, default is 1MB, unit is MB.                                 |
| setPreLoadBufferSize   | Set the pre-play cache size, default is 0.5MB, unit is MB.                              |
| setMaxBufferSize       | Set the video cache size during playback, default is 10MB, unit is MB.                  |
| setPreferredResolution | Set the preferred resolution for video playback, default is 720 x 1280.                 |
| setProgressInterval    | Playback progress callback interval, default is 500 milliseconds, unit is milliseconds. |
| setRenderMode          | Render tiling mode, default is 0. In liteavPlayer, 0 represents full screen, 1          |



|                       | represents rendering according to the actual ratio of the video, which may have black borders.   |
|-----------------------|--|
| setExtInfo            | Set additional information.  |
| setMediaType          | When the media type to be played is known in advance, the media type can be set<br>through this interface to reduce the detection of playback types within the player<br>SDK and improve the startup speed.  |
| enableAutoBitrate     | Set whether to enable bitrate adaptation.  |
| setResumeMode         | Set the resume mode, which is divided into three modes:<br>TUIConstants.TUIResumeMode.NONE: Do not resume playback.<br>TUIConstants.TUIResumeMode.RESUME_LAST: Resume playback from the last<br>time played.<br>TUIConstants.TUIResumeMode.RESUME_PLAYED: Resume playback of all<br>watched videos.  |
| setDisplayViewFactory | Set a custom video layer, which can be customized by implementing IDisplayViewFactory  |
| setEnableAccurateSeek | Set whether to enable precise seeking. After enabling precise seeking, the accuracy of seeking will be greatly improved, but seeking will take time. After disabling it, the actual seeking time may differ from the expected time, depending on the distribution of key frames in the video, but the seeking time will become shorter.  |
| setAudioNormalization | Set volume equalization, with a loudness range of -70 to 0 (LUFS), and also<br>supports custom values. Note: Supported starting from version 11.7 of the advanced<br>player.<br>Predefined values can be filled in, related constant class TXVodConstants,<br>Off: AUDIO_NORMALIZATION_OFF<br>On:<br>AUDIO_NORMALIZATION_STANDARD (standard)<br>AUDIO_NORMALIZATION_LOW (low)<br>AUDIO_NORMALIZATION_HIGH (high) |
| setIsRetainPreVod     | Whether to keep the previous player to speed up the startup of the previous player   |

# 5. Fill in the data

Fill data into TUIShortVideoView:



mSuperShortVideoView.setModels(shortVideoBeanList);

Append data:





mSuperShortVideoView.appendModels(shortVideoBeanList);

#### **TUIVideoSource class**

| Method             | Param Type |  |
|--------------------|------------|--|
| setVideoURL        | String     | Video link, it is recommended to fill in this field, which will speed up the preloading process. |
| setCoverPictureUrl | String     | Video cover, which will be callback to the layer, and  |



|                     |  | handled by the customer.  |
|---------------------|--|---|
| setAppId            | int  | Video appld.  |
| setFileId           | String                                       | Video fileId.   |
| setPSign            | String                                       | Video encryption pSign.   |
| setExtViewType      | int  | Custom page type, this value will be callback through<br>the second parameter of the Layer creation callback,<br>for the business to distinguish different types of pages.  |
| setExtInfoAndNotify | Object                                       | For business to extend additional parameters on their<br>own, using this method can notify messages to the<br>existing layer in real-time. This method will only be<br>valid after obtaining Source through TUIDataManager. |
| setVideoConfig      | TUIPlayerVideoConfig                         | Video independent configuration.  |
| setExternalSubtitle | List <tuisubtitlesource></tuisubtitlesource> | Set external subtitles, which will be loaded into the on-<br>demand player automatically, and must be supported<br>by the advanced version of the player.   |
| setAutoPlay         | boolean                                      | Set whether this video should play automatically.   |

# TUIPlayerVideoConfig class

| Method                   | Param Type | Description   |
|--------------------------|------------|---|
| setPreloadBufferSizeInMB | float      | Set a separate pre-play cache size for the video, optional.             |
| setPreferredResolution   | long       | Set a separate playback and preload resolution for the video, optional. |
| setPreDownloadSize       | float      | Set a separate pre-download cache size for the video, optional.         |

#### TUILiveSource class

| Method             | Param Type | Description  |
|--------------------|------------|--|
| setUrl             | String     | Live streaming link  |
| setCoverPictureUrl | String     | Video Cover, which will be callback to the layer, and handled by the customer. |



| setExtViewType      | int                 | Custom page type, this value will be callback through the second parameter of the Layer creation callback, for the business to distinguish different types of pages.  |
|---------------------|---------------------|---|
| setExtInfoAndNotify | Object              | For business to extend additional parameters on their own,<br>using this method can notify messages to the existing layer<br>in real-time. This method will only be valid after obtaining<br>Source through TUIDataManager. |
| setLiveConfig       | TUIPlayerLiveConfig | Independent configuration.  |
| setAutoPlay         | boolean             | Set whether to play automatically. When this field is set to false for live streaming, there will be no picture at first, and a cover image is needed to cover it.  |

#### TUIPlayerLiveConfig class

| Method          | Param Type | Description  |
|-----------------|------------|--|
| setCacheMinTime | float      | Minimum time for automatic cache<br>adjustment, the value must be<br>greater than 0. [Default Value]: 1.   |
| setCacheMaxTime | float      | Maximum time for automatic<br>cache adjustment, the value must<br>be greater than 0. [Default Value]:<br>5 |

## **TUIPlaySource class**

| Method              | Param Type | Description  |
|---------------------|------------|--|
| setExtViewType      | int        | Custom page type, this value will be callbacked out through<br>the second parameter of the Layer creation callback, for the<br>business to distinguish different types of pages.   |
| setExtInfoAndNotify | Object     | Used for the business to extend additional parameters on<br>their own, this method can notify messages to an existing<br>layer in the interface in real-time. This method will only be<br>valid after being called by a Source obtained through<br>TUIDataManager.<br>6. Operation list data |

TUI Short Video provides data operation interfaces that allow real-time modification of data already added to the list. You can obtain the data operation object by calling the getDataManager() method of TUIShortVideoView, as shown



below:



// Obtain the data operation object
TUIShortVideoDataManager dataManager = mSuperShortVideoView.getDataManager();

After obtaining the data operation object, you can manipulate the list data in real-time. The interfaces are as follows:

| Method     | Returned Param | Passed Param                                  | Description                            |
|------------|----------------|---|--|
| removeData | void           | index: The position of the page to be removed | Remove the corresponding page and data |



| removeRangeData     | void           | index: The starting position of<br>the page to be removed<br>count: The number to be<br>removed from the starting<br>position, excluding the last digit<br>of count | Remove a range of pages and data  |
|---------------------|----------------|---|---|
| removeDataByIndex   | void           | removeIndexList: A collection of positions of the pages to be removed   | Remove all pages and<br>data within the index<br>collection according to<br>the passed index                          |
| addData             | void           | source: The data to be inserted<br>index: The position where the<br>data is to be inserted  | Insert the passed data<br>into the specified<br>position based on the<br>position where the data<br>is to be inserted |
| addRangeData        | void           | sources: A collection of data to<br>be inserted<br>startIndex: The starting position<br>for inserting data  | Insert the data collection<br>into the specified<br>position based on the<br>passed starting position                 |
| replaceData         | void           | source: The data to be replaced index: The position to be replaced  | Replace the data at the specified position with the passed data based on the passed position                          |
| replaceRangeData    | void           | sources: A collection of data to<br>be replaced<br>startIndex: The starting position<br>for replacement   | Replace the specified<br>position with the passed<br>data collection based on<br>the passed starting<br>position      |
| getDataByPageIndex  | TUIVideoSource | index: Page position  | Obtain the page data at<br>the specified position<br>based on the passed<br>position                                  |
| getCurrentDataCount | int            | -   | Get the total number of all data in the current list  |
| getCurrentIndex     | int            | -   | Get the position of the currently displayed page  |
| getCurrentModel     | TUIVideoSource | -   | Get the data of the currently displayed page  |

# 7. Obtain the currently playing video resource

Obtain the currently playing video resource, see the following code for usage:



mSuperShortVideoView.getCurrentModel()

# 8. pause

Pause the currently playing video.





mSuperShortVideoView.pause()

## 9. Play from a specified position.

Start playing from a specified position. This method can directly jump to the specified position during playback, and does not smooth jump by default. See the following code for usage:





// index is the specified page position.
mSuperShortVideoView.startPlayIndex(index);

// index is the position to navigate to, true indicates that smooth switching is re
// and the default is false.
mSuperShortVideoView.startPlayIndex(index, true);

# 10. Set the short video playback mode.

Currently, there are two short video playback modes: list loop playback

TUIVideoConst.ListPlayMode.MODE\_LIST\_LOOP, which automatically plays the next video after the current one finishes, and after playing the last one, it automatically returns to the first video to continue playback. And single video loop playback TUIVideoConst.ListPlayMode.MODE\_ONE\_LOOP, which will keep repeating the current video until the user manually swipes to flip the page. It can also be set to

TUIVideoConst.ListPlayMode.MODE\_CUSTOM, allowing the business to take over the playback logic. When not set, the default is MODE\_ONE\_LOOP. The setting method is as follows:



// set to MODE\_ONE\_LOOP
mSuperShortVideoView.setPlayMode(TUIVideoConst.ListPlayMode.MODE\_ONE\_LOOP);

# 11. Destroy the control.

Destroy the control and resources.



mSuperShortVideoView.release()

## 12. Resume playing the current video.

Resume playing the current video. See the following code for usage:





mSuperShortVideoView.resume()

## **13. Real-time resolution switching**

The TUI short video can switch the current video resolution and the global video resolution in real-time. The interface is as follows:





mSuperShortVideoView.switchResolution(720 \* 1080, TUIConstants.TUIResolutionType.CU

In addition to passing TUIResolutionType, switchType can also directly specify the index of the video that needs to be switched to change the video resolution. The meaning of switchType is as follows:

| Param   | Description                  |
|---------|------------------------------|
| GLOBAL  | Set global resolution        |
| CURRENT | Set current video resolution |
|         |                              |



Other values greater than or equal to 0

Set resolution at specified position

Current priority of resolution, the setting priority of the current video resolution is higher than the global resolution.

# 14. Pause and resume preloading

TUI short video can pause and resume preloading tasks in real-time.



// pause all preload

```
mSuperShortVideoView.pausePreload();
// start preload from current video index
mSuperShortVideoView.resumePreload();
```

When calling to resume preloading, it will continue preloading from the current video onwards.

# 15. Add a layer

TUI short video enables custom UI on the playback interface by adding layers. When the onCreateItemLayer callback occurs, you can utilize the LayerManager provided by the method to add and manage layers. You can inherit from TUIBaseLayer to customize the layers you require. These layers will appear above the VideoView.

The visibility of layers is managed by adding and removing Views, which avoids issues related to excessive rendering.

onCreateVodLayer, onCreateLiveLayer, and onCreateCustomLayer parameters

| Param       | Туре   | Description   |
|-------------|--|---|
| layerManger | TUIVodLayerManager,<br>TUILiveLayerManager<br>And<br>TUICustomLayerManager | Layer management object   |
| viewType    | int  | Current video playback type<br>TUIVideoConst.ItemType.ITEM_TYPE_VOD: On-demand<br>TUIVideoConst.ItemType.ITEM_TYPE_LIVE: Live<br>broadcast<br>TUIVideoConst.ItemType.ITEM_TYPE_CUSTOM: Custom<br>interface<br>Other custom types passed by PlayerSource |



```
layerManger.addLayer(new TUICoverLayer());
layerManger.addLayer(new TUIVideoInfoLayer());
layerManger.addLayer(new TUILoadingLayer());
layerManger.addLayer(new TUIErrorLayer());
```

# 16. Remove the specified layer

Remove the layer from the layer manager.



layerManger.removeLayer(layer);

# 17. Remove all layers

Remove all layers from the layer manager.





layerManger.removeAllLayer();

## 18. Get the current layer level

Pass in the layer to obtain its current level, which can be used to determine the order of layer display and the sequence of touch event propagation.




layerManger.indexOfLayer(layer);

### 19. Custom list scrolling speed

TUI short video provides an interface for customizing the scrolling speed of the list, which can be achieved through the setPageScroller method of TUIShortVideoView. Here's an example:





```
mSuperShortVideoView.setPageScroller(new LinearSmoothScroller(getContext()){
    @Override
    protected float calculateSpeedPerPixel(DisplayMetrics displayMetrics) {
        // Return the time taken for each pixel to slide, measured in milliseconds
        return 25f/displayMetrics.densityDpi;
    }
});
```

## 20. Disable list sliding



You can disable or enable list sliding by using the setUserInputEnabled method of TUIShortVideoView . Here's an example:



// false:disables user sliding
mSuperShortVideoView.setUserInputEnabled(false);

#### 21. Layer callback

Layer callbacks include the base class ITUILayer callbacks, player event notifications, and component events from the videoView. After inheriting from TUIVodLayer , TUILiveLayer , and TUICustomLayer , you can

receive callbacks for video playback according to your functional requirements. Currently, the callback functions are as follows:

## TUIVodLayer class

| Method              | Return Type         | Param  | Description  |
|---------------------|---------------------|--|--|
| isShowing           | boolean             | -  | Whether the current layer is being displayed.  |
| createView          | View                | parent: Layer<br>container                             | Abstract method, needs to be<br>implemented by yourself, used to create<br>the View of the layer.                            |
| tag                 | String              | -  | The tag of the layer, used to distinguish different layers.  |
| unBindLayerManager  | void                | -  | The layer is unbound from the manager,<br>which generally happens when the layer<br>is removed.                              |
| show                | void                | -  | Display the current layer.   |
| hidden              | void                | -  | Hide the current layer.  |
| getView             | T extends View      | -  | Get the View of the current layer.   |
| getVideoView        | TUIBaseVideoView    | -  | Get the current VideoView, if the<br>layerManager has not yet bound with the<br>VideoView, it will return empty.             |
| getPlayerController | TUIPlayerController | -  | Get the current playback Controller, if it has not yet bound with the Controller, it will return empty.                      |
| getPlayer           | ITUIVodPlayer       | -  | Get the current player, if it has not yet<br>bound with the Controller, it will return<br>empty.                             |
| getRenderMode       | int                 | -  | Get the rendering fill mode of the current player screen.  |
| onControllerBind    | void                | controller:<br>Current video<br>playback<br>controller | The current VideoView is bound to the playback controller, that is, the current video becomes the video playing in the list. |
| onControllerUnBind  | void                | controller:  | The VideoView is unbound from the  |



|                       |      | Current video<br>playback<br>controller           | playback controller, which generally indicates that the page has slid out of the interface.  |
|-----------------------|------|---|--|
| onBindData            | void | videoSource:<br>Video data                        | The current VideoView has bound video data.  |
| onViewRecycled        | void | videoView:<br>Current<br>player view<br>container | The current videoView has been recycled.   |
| onExtInfoChanged      | void | videoSource:<br>Changed<br>video source           | When extInfo is set through<br>TUIPlaySource, the layer will notify<br>events through this callback.                                   |
| onShortVideoDestroyed | void | -   | When the entire short video component is destroyed, it will notify the layer through this callback for the layer to release resources. |

## TUILiveLayer class

| Method             | Return Type      | Param                      | Description   |
|--------------------|------------------|----------------------------|---|
| isShowing          | boolean          | -                          | Whether the current layer is currently being displayed.   |
| createView         | View             | parent: Layer<br>container | Abstract method that needs to be implemented by oneself, used for creating the View of the layer. |
| tag                | String           | -                          | The tag of the layer, used to differentiate between different layers.                             |
| unBindLayerManager | void             | -                          | The layer is unbound from the manager,<br>which typically occurs when the layer is<br>removed.    |
| show               | void             | -                          | Show the current layer.   |
| hidden             | void             | -                          | Hide the current layer.   |
| getView            | T extends View   | -                          | Obtain the View of the current layer.   |
| getVideoView       | TUIBaseVideoView | -                          | Get the current VideoView; if the layerManager has not yet bound to the                           |



|                       |                     |  | VideoView, it will return empty.  |
|-----------------------|---------------------|--|---|
| getPlayerController   | TUIPlayerController | -  | Obtain the current playback Controller; if<br>it has not yet bound to the Controller, it<br>will return empty.                              |
| getPlayer             | ITUILivePlayer      | -  | Get the current player; if it has not yet<br>bound to the Controller, it will return<br>empty.  |
| getRenderMode         | int                 | -  | Get the rendering fill mode of the current player screen.   |
| onControllerBind      | void                | controller:<br>Current video<br>playback<br>controller | The current VideoView is bound to the playback controller, meaning the current video becomes the one playing in the list.                   |
| onControllerUnBind    | void                | controller:<br>Current video<br>playback<br>controller | The VideoView is unbound from the playback controller, typically indicating that the page has slid off the interface.                       |
| onBindData            | void                | videoSource:<br>Video data                             | The current VideoView has bound video data.   |
| onViewRecycled        | void                | videoView:<br>Current<br>player view<br>container      | The current videoView has been recycled.  |
| onExtInfoChanged      | void                | videoSource:<br>Changed<br>video source                | After setting extInfo through<br>TUIPlaySource, the layer will notify<br>events through this callback.                                      |
| onShortVideoDestroyed | void                | -  | When the entire short video component is destroyed, it will notify the layer through this callback so that the layer can release resources. |

## TUICustomLayer class

| Method     | Return Type | Param         | Description                                   |
|------------|-------------|---------------|---|
| isShowing  | boolean     | -             | Whether the current layer is being displayed. |
| createView | View        | parent: Layer | Abstract method that needs to be implemented  |



|                       |                   | container   | by oneself, used for creating the View of the layer.  |
|-----------------------|-------------------|---|---|
| tag                   | String            | -   | The tag of the layer, used to distinguish different layers.   |
| unBindLayerManager    | void              | -   | The layer is unbound from the manager, which generally occurs when the layer is removed.  |
| show                  | void              | -   | Display the current layer.  |
| hidden                | void              | -   | Hide the current layer.   |
| getView               | T extends<br>View | -   | Obtain the View of the current layer.   |
| onControllerBind      | void              | -   | The current VideoView is bound to the playback controller, meaning the current video is displayed in the list.                              |
| onControllerUnBind    | void              | -   | The VideoView is unbound from the playback controller, typically indicating that the page has slid off the interface.                       |
| onBindData            | void              | videoSource:<br>Video data                        | The current VideoView has bound video data.   |
| onViewRecycled        | void              | videoView:<br>Current<br>player view<br>container | The current videoView has been recycled.  |
| onExtInfoChanged      | void              | videoSource:<br>Changed<br>video source           | After setting extInfo through TUIPlaySource,<br>the layer will notify events through this<br>callback.                                      |
| onShortVideoDestroyed | void              | -   | When the entire short video component is destroyed, it will notify the layer through this callback so that the layer can release resources. |

#### **TUIVodObserver class**

| Method        | Return<br>Type | Param | Description     |
|---------------|----------------|-------|-----------------|
| onPlayPrepare | void           | -     | Video is ready. |

| onPlayBegin                   | void | -  | Video starts playing.   |
|-------------------------------|------|--|---|
| onPlayPause                   | void | -  | Video pauses.   |
| onPlayStop                    | void | -  | Video stops.  |
| onPlayLoading                 | void | -  | Video begins loading.   |
| onPlayLoadingEnd              | void | -  | Video loading completes.  |
| onPlayProgress                | void | current: The current video<br>playback progress, in<br>milliseconds, of type long.<br>duration: The total duration<br>of the current video, in<br>milliseconds, of type long.<br>playable: The playable<br>duration of the current<br>video, in milliseconds, of<br>type long. | Video playback progress.  |
| onSeek                        | void | position: The progress of<br>the video to jump to. In<br>seconds, of type int.   | Video progress seeked.  |
| onError                       | void | code: Video error code.<br>message: Error description.   | An error occurred during video playback.  |
| onRcvFirstIframe              | void | -  | Received the first frame event.   |
| onRcvTrackInformation         | void | infoList: Video tracks.  | Received video track information.   |
| onRcvSubTitleTrackInformation | void | infoList: Video subtitle information.  | Received video subtitle information.  |
| onRecFileVideoInfo            | void | params: Video file<br>information.   | Received video file<br>information, which<br>generally only triggers this<br>callback when playing<br>using fileId. |
| onResolutionChanged           | void | width: Video width.<br>height: Video height.   | The current video resolution changes.   |
| onPlayEvent                   | void | player: Player.<br>event: Event ID.  | All event notifications from the player.  |



|           |      | bundle: Event content. |                                       |
|-----------|------|------------------------|---------------------------------------|
| onPlayEnd | void | -                      | The current video playback has ended. |

#### **TUILiveObserver class**

| Medtho                   | Param   | Description                          |
|--------------------------|---|--------------------------------------|
| onError                  | player: Current live broadcast<br>player<br>code: Error code<br>msg: Error message<br>extraInfo: Additional<br>message  | An error occurred during playback.   |
| onWarning                | player: Current live broadcast<br>player<br>code: Error code<br>msg: Warning message<br>extraInfo: Additional<br>message  | A warning occurred in the player.    |
| onVideoResolutionChanged | player: Current live broadcast<br>player<br>width: Video width<br>height: Video height  | The player's resolution has changed. |
| onConnected              | player: Current live broadcast<br>player<br>extraInfo: Additional<br>information  | Data stream connection successful.   |
| onVideoPlaying           | player: Current live broadcast<br>player<br>firstPlay: Whether it is the first<br>playback, <b>which can be</b><br><b>used to determine the first</b><br><b>frame</b><br>extraInfo: Additional<br>information | Video playback has started.          |
| onAudioPlaying           | player: Current live broadcast<br>player<br>firstPlay: Whether it is the first<br>playback  | Audio playback has started.          |



|                       | extraInfo: Additional information  |   |
|-----------------------|--|---|
| onVideoLoading        | player: Current live broadcast<br>player<br>extraInfo: Additional<br>information   | Video loading has begun.  |
| onAudioLoading        | player: Current live broadcast<br>player<br>extraInfo: Additional<br>information   | Audio loading has begun.  |
| onPlayoutVolumeUpdate | player: Current live broadcast<br>player<br>volume: Changed volume   | Player volume size callback.  |
| onStatisticsUpdate    | player: Current live broadcast<br>player<br>statistics: Data details   | Live broadcast player statistics callback.  |
| onSnapshotComplete    | player: Current live broadcast<br>player<br>image: Screenshot image  | Screenshot callback.  |
| onRenderVideoFrame    | player: Current live broadcast<br>player<br>videoFrame: Image frame  | Custom video rendering callback, enabled by calling enableObserveVideoFrame.                      |
| onPlayoutAudioFrame   | player: Current live broadcast<br>player<br>audioFrame: Audio frame  | Custom audio data callback, enabled by calling enableObserveAudioFrame.                           |
| onReceiveSeiMessage   | player: Current live broadcast<br>player<br>payloadType: SEI<br>payloadType of the callback<br>data.<br>data: Data                                   | Callback for receiving SEI messages, sent by the sender through sendSeiMessage in V2TXLivePusher. |
| onStreamSwitched      | player: Current live broadcast<br>player<br>url: Switched URL<br>code: Status code, 0:<br>Success, -1: Switching timed<br>out, -2: Switching failed, | Resolution seamless switching callback.   |



|                       | server error, -3: Switching failed, client error  |  |
|-----------------------|---|--|
| onLocalRecordBegin    | <ul> <li>player: Current live broadcast</li> <li>player</li> <li>code: Status code.</li> <li>0: Recording task started</li> <li>successfully.</li> <li>-1: Internal error caused</li> <li>recording task startup failure.</li> <li>-2: Incorrect file extension</li> <li>(such as unsupported</li> <li>recording format).</li> <li>-6: Recording has already</li> <li>started, need to stop</li> <li>recording first.</li> <li>-7: Recording file already</li> <li>exists, need to delete the file</li> <li>first.</li> <li>-8: Recording directory has</li> <li>no write permission, please</li> <li>check the directory</li> <li>permission issue.</li> <li>storagePath: Recorded file</li> <li>address.</li> </ul> | Event callback for the start of recording tasks.         |
| onLocalRecording      | player: Current live broadcast<br>player<br>durationMs: Recording<br>duration<br>storagePath: Recorded file<br>address.   | Progress event callback for recording tasks in progress. |
| onLocalRecordComplete | <ul> <li>player: Current live broadcast</li> <li>player</li> <li>code: Status code.</li> <li>0: End recording task</li> <li>successfully.</li> <li>-1: Recording failed.</li> <li>-2: Switching resolution or</li> <li>landscape/portrait screen</li> <li>caused recording to end.</li> <li>-3: Recording time is too</li> <li>short, or no video or audio</li> <li>data has been collected,</li> <li>please check the recording</li> </ul>   | Event callback for recording tasks that have ended.      |

|  | duration, or whether audio<br>and video collection has<br>been enabled.<br>storagePath: Recorded file<br>address. |  |  |
|--|---|--|--|
|--|---|--|--|

## 22. Player functions

In the layer, you can obtain the player object through getPlayer. The interface functions of the player ITUIVodPlayer and ITUILivePlayer are as follows:

#### ITUIVodPlayer class

| Medthod            | Return Type | Param   | Descrij  |
|--------------------|-------------|---|--|
| prePlay            | void        | model: Video data, type TUIVideoSource                                    | Preload<br>video, w<br>deduplic<br>mechar<br>inside. |
| resumePlay         | void        | -   | Continu<br>playing<br>current                        |
| seekTo             | void        | time: The time point to jump to, in seconds, of type int.                 | Jump tc<br>specifie<br>playbac<br>position           |
| isPlaying          | boolean     | -   | Check i<br>current<br>is playir                      |
| startPlay          | void        | model: Video data, type TUIVideoSource.                                   | Play the video.                                      |
| pause              | void        | -   | Pause t<br>playbac                                   |
| stop               | void        | needClearLastImg: Optional parameter, whether to clear the current image. | Stop the playbac                                     |
| getCurrentPlayTime | float       | -   | Get the<br>current<br>playbac                        |



|                      |      |  | time, in<br>second:   |
|----------------------|------|--|---|
| setDisplayView       | void | videoView: Video rendering View, type<br>TUIVideoRenderView.   | Set the<br>to be<br>rendere<br>the play   |
| setSurface           | void | surface : render surface   | Set the<br>surface<br>rendere<br>the vide   |
| addPlayerObserver    | void | observer: Player observer, type<br>TUIVodObserver.   | Set the observe   |
| removePlayerObserver | void | observer: Player observer, type<br>TUIVodObserver.   | Remove<br>player<br>observe   |
| setConfig            | void | config: Video configuration, type<br>TXVodPlayConfig.  | Set the configur  |
| setRenderMode        | void | renderMode: Rendering mode,<br>FULL_FILL_SCREEN: Maintain aspect<br>ratio and fill the screen<br>ADJUST_RESOLUTION: Maintain<br>aspect ratio and display the video | Set the mode.   |
| setStartTime         | void | startTime: Start playing time, in seconds.<br>float type   | Set the<br>playbac<br>time, va<br>before<br>playbac<br>takes el<br>once. Lo<br>playbac<br>start fro<br>beginnii |
| setLoop              | void | isLoop: Whether to loop  | Set whe<br>the vide<br>looped.  |
| setBitrateIndex      | void | index: Bitrate index   | Set the current bitrate.  |



| switchResolution       | void   | resolution: Resolution, i.e., width × height | Set the<br>player's<br>resolution<br>this play<br>if there in<br>corresp<br>resolution<br>will be<br>switche<br>only tak<br>effect fo<br>playbac |
|------------------------|--|--|--|
| getSupportResolution   | List <tuiplayerbitrateitem></tuiplayerbitrateitem> | -  | Get the<br>resolution<br>the vide<br>being pl  |
| getBitrateIndex        | int  | -  | Get the<br>index of<br>video be<br>played.   |
| setRate                | void   | rate: Video speed, normal speed is 1.0       | Set the<br>playbac<br>speed c<br>current   |
| getCurrentPlaybackTime | float  | _  | Get the<br>already<br>played I<br>current<br>in secor  |
| getBufferDuration      | float  | -  | Get the<br>bufferec<br>of the cu<br>video, ir<br>seconds   |
| getDuration            | float  | -  | Get the<br>duratior<br>current<br>in secor   |
| getPlayableDuration    | float  | -  | Get the  |



|                       |         |  | playable<br>of the ci<br>video, ir<br>second:                    |
|-----------------------|---------|--|--|
| getWidth              | int     | -  | Get the<br>of the vi<br>being pl                                 |
| getHeight             | int     | -  | Get the<br>of the vi<br>being pl                                 |
| setRenderRotation     | void    | rotation: Angle  | Set the<br>rotation<br>of the ci<br>video.                       |
| enableHardwareDecode  | boolean | enable: Whether to enable hardware decoding                                | Set whe<br>the curr<br>player u<br>hardwa<br>decodin             |
| setMute               | void    | mute: Whether to mute  | Set whe<br>the curr<br>playing<br>is muter                       |
| setAudioPlayoutVolume | void    | volume: Video volume, 0 to 100   | Set the volume current   |
| setRequestAudioFocus  | boolean | requestFocus: Whether to get audio focus                                   | Set whe<br>the curr<br>player g<br>audio fc                      |
| snapshot              | void    | listener:<br>TXLivePlayer.ITXSnapshotListener type,<br>screenshot callback | Take a<br>screens<br>the vide<br>being pl<br>by the c<br>player. |
| setMirror             | void    | mirror: Whether to mirror  | Set whe the curr   |



|                   |         |  | video is<br>mirrorec   |
|-------------------|---------|--|--|
| setToken          | void    | token: Encrypted HLS token   | Set the<br>encrypt<br>HLS tok  |
| isLoop            | boolean | -  | Check i<br>current<br>is in loo<br>playbac   |
| attachTRTC        | void    | trtcCloud: trtc service object   | Push th<br>audio ai<br>video st<br>of the pl<br>through<br>TRTC, i<br>TRTC<br>services<br>be founi<br>TRTC p<br>overviei |
| detachTRTC        | void    | -  | Unbind<br>service<br>the play  |
| setStringOption   | void    | key: Business parameter key value<br>value: Business parameter value   | Set the<br>busines<br>parame<br>the play   |
| setSubtitleView   | void    | subtitleView: Subtitle component                                       | Set the<br>subtitle<br>compor<br>the vide  |
| addSubtitleSource | void    | url: Subtitle link<br>name: Subtitle name<br>mimeType: Subtitle format | Add sut<br>sources<br>video.   |
| selectTrack       | void    | trackIndex: Audio/video track  | Add/sel<br>audio/vi<br>tracks,<br>commoi<br>used foi   |



|                       |                                  |   | audio tr<br>and sub<br>selectio                        |
|-----------------------|----------------------------------|---|--|
| deselectTrack         | void                             | trackIndex: Audio/video track   | Remove<br>selected<br>audio/vi<br>tracks.              |
| setAudioNormalization | void                             | value: Loudness range: -70 to 0 (LUFS),<br>with custom values supported. Note: The<br>player advanced version 11.7 starts to<br>support this feature. Fill in the preset<br>value, related constants<br>TXVodConstants,<br>Off: AUDIO_NORMALIZATION_OFF<br>On:<br>AUDIO_NORMALIZATION_STANDARD<br>(Standard)<br>AUDIO_NORMALIZATION_LOW (Low)<br>AUDIO_NORMALIZATION_HIGH<br>(High) | Set volu<br>equaliza                                   |
| setSubtitleStyle      | void                             | renderModel: Subtitle format  | Set sub<br>format.                                     |
| getSubtitleTrackInfo  | List <txtrackinfo></txtrackinfo> | -   | Get imp<br>subtitle<br>informa<br>valid aft<br>prepare |
| getAudioTrackInfo     | List <txtrackinfo></txtrackinfo> | -   | Get imp<br>audio tr<br>informa<br>valid aft<br>prepare |

## TUIPlayerBitrateItem class

| Method   | Return Type | Description                       |
|----------|-------------|-----------------------------------|
| getIndex | int         | Current resolution bitrate index. |
| getWidth | int         | Current resolution video width.   |



| getHeight  | int | Current resolution video height.  |
|------------|-----|-----------------------------------|
| getBitrate | int | Current resolution video bitrate. |

## ITUILivePlayer class

| Medthod                | Return Type   | Param   | De               |
|------------------------|---|---|------------------|
| prePlay                | void  | model: Video data, type<br>TUILiveSource  | Pr∉<br>m∈        |
| resumePlay             | void  | -   | Со               |
| setConfig              | void  | config: Live streaming configuration  | Se               |
| addLiveObserver        | void  | observer: Live event<br>observer  | Su               |
| removeLiveObserver     | void  | observer: Live event<br>observer  | Un               |
| switchStream           | int   | model: The data source<br>to be switched  | Sw<br>cui<br>su( |
| setRenderRotation      | int   | rotation: Rotation<br>direction   | Se               |
| setPlayoutVolume       | int   | volume: Volume size,<br>ranging from 0 to 100.<br>【Default】: 100  | Se               |
| getStreamList          | ArrayList <v2txlivedef.v2txlivestreaminfo></v2txlivedef.v2txlivestreaminfo> | -   | Ge               |
| enableVolumeEvaluation | int   | value: Determines the<br>trigger interval of the<br>onPlayoutVolumeUpdate<br>callback, in milliseconds,<br>with a minimum interval<br>of 100ms. If it is less<br>than or equal to 0, the<br>callback will be turned<br>off, and it is<br>recommended to set it to<br>300ms; [Default] : 0,<br>not enabled | En               |



| snapshot                | int  | -   | Ca<br>Re<br>V<br>V<br>is i<br>sci          |
|-------------------------|------|---|--|
| enableReceiveSeiMessage | int  | enable: true: Enable<br>receiving SEI messages;<br>false: Disable receiving<br>SEI messages.<br>[Default] : false.<br>payloadType: Specify<br>the payloadType to<br>receive SEI messages,<br>supporting 5 and 242,<br>and keep it consistent<br>with the payloadType of<br>the sending end. | En   |
| showDebugView           | void | isShow: Whether to<br>display.【Default】:<br>false.  | Wł<br>pla                                  |
| setProperty             | int  | key: The key<br>corresponding to the<br>advanced API.<br>value: The parameter<br>needed when calling the<br>advanced API<br>corresponding to the<br>key.  | Ca<br>V2                                   |
| startLocalRecording     | int  | params: Local recording<br>audio and video<br>configuration   | Sta<br>Re<br>V2<br>V2<br>: In<br>V2<br>ref |
| stopLocalRecording      | void | -   | Stc<br>stc<br>rec<br>rec                   |

## Advanced features

## 1. Business notification messages to page layers

TUI provides a message interface for users to notify data in real-time to the current layer. After obtaining the video object through the data operation object, notifications can be made. The example is as follows:



```
// get data controller
TUIShortVideoDataManager dataManager = mSuperShortVideoView.getDataManager();
// get dataSource
TUIPlaySource videoSource = dataManager.getDataByPageIndex(0);
```

## 🕗 Tencent Cloud

```
Map<String, String> data = new HashMap<>();
data.put("key1", "data1");
// set extInfo and notify to layer
videoSource.setExtInfoAndNotify(data);
```

Subsequently, the notification will be received in the layer's onExtInfoChanged callback, allowing UI modifications to the current page. The example is as follows:



@Override
public void onExtInfoChanged(TUIVideoSource videoSource) {
 super.onExtInfoChanged(videoSource);

```
Map<String, String> data = (Map<String, String>) videoSource.extInfo;
Log.i("tag", "get data:" + data);
}
```

#### Note:

onExtInfoChanged will only be triggered after binding data with onBindData .

#### 2. Set volume balance for on-demand playback

The player supports automatic volume adjustment while playing audio to ensure that the volume of all audio is consistent. This can avoid issues where some audios are too loud or too quiet, providing a better listening experience. By setting volume balance, the loudness range is -70 to 0 (LUFS), and custom values are also supported.

#### Note:

Supported starting from version 11.7 of the premium player edition.



| /*  |
|---|
| * Supports custom numerical values, loudness range: -70 to 0 (LUFS)   |
| * Preset values: Off: TXVodConstants#AUDIO_NORMALIZATION_OFF          |
| * On (Low): TXVodConstants#AUDIO_NORMALIZATION_LOW                    |
| * On (Standard): TXVodConstants#AUDIO_NORMALIZATION_STANDARD          |
| * On (High): TXVodConstants#AUDIO_NORMALIZATION_HIGH                  |
| */  |
| TUIPlayerVodStrategy vodStrategy = new TUIPlayerVodStrategy.Builder() |
| .setAudioNormalization(TXVodConstants.AUDIO_NORMALIZATION_STANDARD)   |
| .build();   |
| <pre>mShortVideoView.setVodStrategy(vodStrategy);</pre>               |



# API Documentation Web

Last updated : 2024-06-06 10:34:01

This document is an introduction to the parameters and API of the Web Player (TCPlayer) suitable for live and ondemand playback. It is intended for developers with a basic understanding of JavaScript.

## Parameters initialization

Initializing the player requires two parameters: the first is the player's container ID, and the second is the feature parameter object.





var player = TCPlayer('player-container-id', options);

#### **Options parameter list**

The configurable parameters for the options object are as follows:

| Name  | Туре   | Default Value | Description  |
|-------|--------|---------------|--|
| appID | String | No            | Mandatory when playing on-demand media files through fileID, as it corresponds to the appID of the Tencent Cloud account |



| fileID        | String        | No                 | Mandatory when playing on-demand media files<br>through fileID, as it is the ID of the on-demand media<br>file   |
|---------------|---------------|--------------------|--|
| psign         | String        | No                 | Player Signature, required when playing through fileID. For more details, see Player Signature   |
| licenseUrl    | String        | No                 | Player License address, peek at Player License   |
| sources       | Array         | No                 | Player playback address, format: [{ src:<br>'//path/to/video.mp4', type: 'video/mp4' }]  |
| width         | String/Number | No                 | Player zone width, in pixels, set as needed. The player size can be controlled through CSS.  |
| height        | String/Number | No                 | Player zone height, in pixels, set as needed. The player size can be controlled through CSS.   |
| controls      | Boolean       | true               | Whether to show the player's control bar.  |
| poster        | String        | No                 | Set the full address of the cover image (if the<br>uploaded video has a generated cover image, it will<br>be used preferentially. For details, please see Video<br>on Demand - Manage Videos). |
| autoplay      | Boolean       | false              | Whether to autoplay.   |
| playbackRates | Array         | [0.5,1,1.25,1.5,2] | Set the adjustable-speed playback options, only valid in HTML5 playback pattern.   |
| loop          | Boolean       | false              | Whether to loop playback.  |
| muted         | Boolean       | false              | Whether to mute playback.  |
| preload       | String        | auto               | Whether preloading is needed, with 3 attributes:<br>"auto", "meta", and "none". Due to system<br>restrictions on mobile devices, setting auto is<br>ineffective.                               |
| swf           | String        | No                 | URL of the Flash player's SWF file   |
| posterImage   | Boolean       | true               | Whether to display the cover.  |
| bigPlayButton | Boolean       | true               | Whether to display a centered play button (the browser hijacked embedded play button cannot be removed).   |
| language      | String        | "zh-CN"            | Setting language, the options are "zh-CN"/"en"   |



| languages      | Object  | No    | Setting multilingual dictionary.  |
|----------------|---------|-------|---|
| controlBar     | Object  | No    | Setting the parameter combination for control bar properties, see controlBar parameter list.        |
| reportable     | Boolean | true  | Setting whether to enable data reporting.   |
| fakeFullscreen | Boolean | false | Setting up pseudo-full screen through style control to achieve the full screen effect.              |
| plugins        | Object  | No    | Setting the parameter combination for plugin feature attributes, see plugins plugin parameter list. |
| hlsConfig      | Object  | No    | HLS.js startup configuration, for detailed content, please see the official documentation hls.js.   |
| webrtcConfig   | Object  | No    | WebRTC startup configuration, see webrtcConfig parameter list.                                      |
| xp2pConfig     | Object  | No    | P2P startup configuration, see xp2pConfig<br>parameter list. For P2P feature details, see X-P2P.    |

#### Note:

The parameters controls, playbackRates, loop, preload, posterImage will be ineffective when the browser hijacks playback.

For issues with browser hijacking video playback, see FAQs Description.

#### controlBar parameter list

controlBar parameters can configure the player's control bar features, with supported properties as listed in the following table:

| Name               | Туре    | Default<br>Value | Description                                      |
|--------------------|---------|------------------|--|
| playToggle         | Boolean | true             | Whether to display the play/pause toggle button. |
| progressControl    | Boolean | true             | Whether to display the progress bar.             |
| volumePanel        | Boolean | true             | Whether to display volume control.               |
| currentTimeDisplay | Boolean | true             | Whether to show the current time of the video.   |
| durationDisplay    | Boolean | true             | Whether to display the video duration.           |
|                    |         |                  |  |



| timeDivider               | Boolean | true | Whether to display the time separator.                  |
|---------------------------|---------|------|---|
| playbackRateMenuButton    | Boolean | true | Whether to display the playback speed selection button. |
| fullscreenToggle          | Boolean | true | Whether to display the full-screen button.              |
| QualitySwitcherMenuButton | Boolean | true | Whether to display the definition switch menu.          |

#### **Please Note:**

The controlBar parameter will be ineffective when the browser hijacks playback.

For issues with browser hijacking video playback, see FAQs Description.

#### plugins plugin parameter list

plugins parameters can configure the player's plugin features, supported properties include:

| Name             | Туре    | Default<br>Value | Description  |
|------------------|---------|------------------|--|
| ContinuePlay     | Object  | No               | Controlling the resume playback feature, supported properties are a auto:Boolean Whether it automatically resumes playback.<br>text:String Prompt text.<br>btnText:String Button text.   |
| VttThumbnail     | Object  | No               | Controlling thumbnail display, supported properties are as follows:<br>vttUrl:String Absolute address of vtt file, required.<br>basePath:String Image path, optional. If not provided, the path of vt<br>imgUrl:String Absolute image address, optional.   |
| ProgressMarker   | Boolean | No               | Controlling the display of the progress bar.   |
| DynamicWatermark | Object  | No               | Controlling the display of dynamic watermarks, supporting text and<br>are:<br>type:String Watermark type can be text or image, with values text  <br>content:String Text watermark content, required.<br>speed:Number Watermark movement speed, value range 0-1, defa<br>opacity:Number Text watermark opacity, value range 0-1, optional.<br>fontSize:String Font size, default 12px, optional.<br>color: String Text color, optional.<br>left:String Text position, supports percentage and px units. When so<br>ineffective, optional.<br>top,right,bottom: Description same as left.<br>width:String Image watermark width, optional.<br>height:String Image watermark height, optional. |
| ContextMenu      | Object  | No               | Valid values are as follows:   |











| VR        | Object | No | Advanced Edition License support, for details see Web advanced for (TCPlayerVRPlugin)        |
|-----------|--------|----|--|
| SafeCheck | Object | No | Advanced Edition License support, for details see Web advanced for (TCPlayerSafeCheckPlugin) |

#### webrtcConfig parameter list

webrtcConfig parameters control the behavior during WebRTC playback, with the following attributes supported:

| Name              | Туре    | Default Value | Description  |
|-------------------|---------|---------------|--|
| connectRetryCount | Number  | 3             | Number of reconnections between the SDK and the server |
| connectRetryDelay | Number  | 1             | Delay in reconnection between SDK and server           |
| receiveVideo      | Boolean | true          | Whether to fetch video stream                          |
| receiveAudio      | Boolean | true          | Whether to fetch audio stream                          |
| showLog           | Boolean | false         | Should logs be printed in the console                  |

#### xp2pConfig parameter list

Before using X-P2P, you need to apply for activation. Please go to X-P2P and click to apply. After applying, our dedicated product support staff will contact you.

For more detailed information, please refer to the X-P2P Product Documentation.

#### **Common Parameters**

| Name              | Туре    | Default Value | Description   |
|-------------------|---------|---------------|---|
| useXP2P           | Boolean | false         | Whether to enable XP2P  |
| format            | String  | No            | Specify the media protocols that P2P needs<br>to support based on the current video format.<br>The valid values are as follows:<br>flv<br>hls<br>webrtc |
| tencentCloudAppId | Number  | No            | In your Tencent Cloud account's APPID<br>(Console peek path: Account Center ><br>Account Information > Basic<br>Information > APPID)                    |
|                   |         |               |   |

| xp2pAppId   | String | No | X-P2P assigned ID, provided by our email      |
|-------------|--------|----|---|
| хр2рАррКеу  | String | No | X-P2P assigned Key, provided by our email     |
| xp2pPackage | String | No | X-P2P assigned Package, provided by our email |

## Additional parameters for the flv protocol

| Name           | Туре    | Default Value | Description   |
|----------------|---------|---------------|---|
| bizId          | String  | No            | Provided by our email                                     |
| xp2pPlayDomain | String  | No            | FLV protocol streaming domain name, provided by our email |
| authMode       | String  | No            | Authentication pattern, provided by our email             |
| debug          | Boolean | false         | Debug switch  |

## Additional parameters for the HLS protocol

| Name                | Туре    | Default Value           | Description   |
|---------------------|---------|-------------------------|---|
| videoType           | String  | VOD                     | Please specify whether the current HLS<br>playback is live or on-demand accurately.<br>The valid values are as follows:<br>LIVE<br>VOD  |
| channelld           | String  | Automatically generated | You can proactively generate an ID for the<br>current resource. If not specified, it defaults<br>to internal automatic generation. See the<br>generation rules as follows HLS Resource ID<br>Generation Rules |
| channelldWithHost   | Boolean | true                    | Defaults to true, with an optional<br>configuration, usually no need to modify this<br>configuration. For a detailed explanation, see<br>the following HLS Resource ID Generation<br>Rules                    |
| channelldWithSearch | Boolean | false                   | Defaults to false, with an optional<br>configuration, usually no need to modify this<br>configuration. For a detailed explanation, see<br>the following HLS Resource ID Generation<br>Rules                   |

#### **HLS Resource ID Generation Rules**

The Resource ID is the unit of P2P sharing; only nodes with the same Resource ID can share with each other via P2P. Different videos must ensure different Resource IDs, otherwise, streams will be crossed.

#### 1.1 Proactively Input

By using the Setting parameter channelId field, you can proactively assign a Resource ID for the current video, ensuring it uniquely identifies the file to avoid stream crossing.

#### 1.2 Default Generation

If the channelld field is not input, the sdk will by default generate an ID for each url; IDs that are the same will share with each other via P2P. The ID generation rule is as follows:

(Default) Intercept the host and path parts to generate an MD5 checksum.

For example: https://a.b.com/p1/p2/p3.m3u8?m=1&n=2, then the ID = MD5('a.b.com/p1/p2/p3.m3u8')

#### 1.3 Input parameters control the default generation rule

(Optional, default is true) By entering the channelIdWithHost parameter, true indicates that the ID includes the host part.

(Optional, default is false) By entering the channelIdWithSearch parameter, false indicates it includes the search part.

For example: https://a.b.com/p1/p2/p3.m3u8?m=1&n=2

If { channelIdWithHost: true, channelIdWithSearch: true } is passed, then the ID =

MD5('a.b.com/p1/p2/p3.m3u8?m=1&n=2')

If { channelIdWithHost: false, channelIdWithSearch: false } is passed, then the ID = MD5('/p1/p2/p3.m3u8')

#### Note:

You can generate different Resource IDs for video URLs that cannot share P2P, and the same Resource ID for URLs that can, based on your own business URL generation rules.

#### 1.4 Multi-bitrate M3U8 Explanation

If the video being played is multi-bitrate M3U8, we ensure that nodes of different bitrates do not P2P with each other.

#### **X-P2P Protocol Support**

| Audio/Video Use Protocol | PC Browser | Android | iOS |
|--------------------------|------------|---------|-----|
|--------------------------|------------|---------|-----|



| FLV    | Live stream                           | Supported<br>chrome 55+<br>firefox 65+<br>safari 11+ | Supported<br>chrome 55+<br>firefox 65+<br>Weixin browser | Not supported |
|--------|---------------------------------------|--|--|---------------|
| HLS    | Live streaming,<br>video on<br>demand | Supported<br>chrome 55+<br>firefox 65+<br>safari 11+ | Supported<br>chrome 55+<br>firefox 65+<br>Weixin browser | Not supported |
| WebRTC | Live stream                           | Partially supported<br>Chrome 94+<br>edge 94+        | Partially supported<br>Chrome 94+<br>edge 94+            | Not supported |

## Object methods

Methods list returned by initialize player object:

| Name                 | Parameters and Types        | Return Value and Types                 | Description   |
|----------------------|-----------------------------|--|---|
| src()                | (String)                    | No                                     | Setting the playback address.   |
| loadVideoByID()      | (Object)                    | No                                     | When playing with fileID, you can switch videos using this method. The parameter is an object consisting of fileID, appID, psign. |
| ready(function)      | (Function)                  | No                                     | Setting the callback after player initialization is complete.   |
| play()               | No                          | No                                     | Play and resume playback.   |
| pause()              | No                          | No                                     | Pauses playback.  |
| currentTime(seconds) | (Number)                    | (Number)                               | Access the current playback time, or set<br>the playback time, which cannot exceed<br>the video duration.                         |
| duration()           | No                          | (Number)                               | Access video duration.  |
| volume(percent)      | (Number)[0,1]<br>[Optional] | (Number)/ No<br>return when<br>setting | Access or set the player volume.  |
| muted()              | (Boolean)                   | (Boolean)                              | Access or set whether the player is muted   |



| playbackRate()      | (Number)[0, 1]         | (Number)                               | Access or set the playback speed  |
|---------------------|------------------------|--|---|
| poster(src)         | (String)               | (String)/ No return when setting       | Access or set the player thumbnail.   |
| requestFullscreen() | No                     | No                                     | Enter full screen mode pattern.   |
| exitFullscreen()    | No                     | No                                     | Exit full screen mode pattern.  |
| isFullscreen()      | No                     | Boolean                                | Return whether full screen mode pattern has been entered.   |
| on(type,listener)   | (String,<br>Function)  | No                                     | Listening to events.  |
| one(type,listener)  | (String,<br>Function)  | No                                     | Listening to events, the event handler executes no more than once.  |
| off(type,listener)  | (String,<br>Function)  | No                                     | Unbind event listener.  |
| buffered()          | No                     | TimeRanges                             | Returns the video buffer range.   |
| bufferedPercent()   | No                     | Value range [0,1]                      | Returns the buffer length as a percentage of video duration.  |
| width()             | (Number)<br>[Optional] | (Number)/ No<br>return when<br>setting | Access or set the player zone width. If the player size is set via CSS, this method will be ineffective.        |
| height()            | (Number)<br>[Optional] | (Number)/ No<br>return when<br>setting | Access or set the player zone height. If<br>the player size is set via CSS, this<br>method will be ineffective. |
| videoWidth()        | No                     | (Number)                               | Access video resolution width.  |
| videoHeight()       | No                     | (Number)                               | Access video resolution height.   |
| dispose()           | No                     | No                                     | Terminate the player.   |

#### Note

Object methods cannot be called synchronously; they need to be called after the corresponding event (such as loadedmetadata) is triggered, except for ready, on, one, and off.

## Event


The player can perform event listening through the objects returned by the initialization, for example:



```
var player = TCPlayer('player-container-id', options);
// player.on(type, function);
player.on('error', function(error) {
    // Do some processing
});
```

Among them, type is the event type, supported events include:

| Name | Features |
|------|----------|
|      |          |



| play                | Playback has started, triggered by calling the play() method or setting autoplay to true and it takes effect, with the paused attribute being false.  |
|---------------------|---|
| playing             | Triggered when playback resumes after being paused or stopped for buffering, with the paused attribute being false. This event is commonly used to mark the real start of video playback, as the play event only indicates the initiation of playback without actual video rendering. |
| loadstart           | Triggered when starting to load data.   |
| durationchange      | Triggered when the video's duration data changes.   |
| loadedmetadata      | The video's metadata has been loaded.   |
| loadeddata          | This event is triggered when the data for the current frame is loaded, but there is not enough data to play the next frame of the video.  |
| progress            | Triggered when accessing media data.  |
| canplay             | Triggered when the player is able to start playing the video.   |
| canplaythrough      | Triggered when the player estimates it can play through the specified video without stopping for buffering.   |
| error               | Triggered when an error occurs during video playback.   |
| pause               | Triggered when paused.  |
| blocked             | Triggered when autoplay is blocked by the browser.  |
| ratechange          | Triggered when the playback speed changes.  |
| seeked              | Triggered when the search for a specified playback position ends.   |
| seeking             | Triggered when the search for a specified playback position begins.   |
| timeupdate          | Triggered when there is a change in the current playback position, which can be understood as a change in currentTime.  |
| volumechange        | Triggered when the volume setting or muted property value changes.  |
| waiting             | Triggered when playback stops, and the next frame of content is unavailable.  |
| ended               | Triggered when video playback has ended. At this point, the value of currentTime is equal to the maximum value of the media resource.   |
| resolutionswitching | Definition switch in progress.  |
| resolutionswitched  | Definition switch completed.  |
|                     |   |



| fullscreenchange | Triggered during full screen status toggle.  |  |
|------------------|--|--|
| webrtcevent      | Event set during webrtc playback.            |  |
| webrtcstats      | Statistics during webrtc playback.           |  |
| webrtcfallback   | Triggered degradation during webrtc playback |  |

### WebrtcEvent list

The player can access all events during webrtc playback through webrtcevent, for example:



### 🕗 Tencent Cloud

```
var player = TCPlayer('player-container-id', options);
player.on('webrtcevent', function(event) {
    // Access event status code and related data from the callback parameter event
});
```

| webricevent status codes are as follows | webrtcevent status | codes | are | as | follows |
|---|--------------------|-------|-----|----|---------|
|---|--------------------|-------|-----|----|---------|

| Status Code | Callback parameters | Features   |
|-------------|---------------------|--|
| 1001        | No                  | Start pulling stream   |
| 1002        | No                  | Connected to server  |
| 1003        | No                  | Video playback starts  |
| 1004        | No                  | Stop pulling stream, end video playback                          |
| 1005        | No                  | Failed to connect to server. Auto-reconnect initiated to recover |
| 1006        | No                  | Access stream data is empty                                      |
| 1007        | localSdp            | Starting signaling server request                                |
| 1008        | remoteSdp           | Signaling server request succeeded                               |
| 1009        | No                  | Streaming lagging, buffering in progress                         |
| 1010        | No                  | Streaming lag ended, playback resumed                            |

### Error code

When the player triggers an error event, the listener function returns an error code, with those having three or more digits being media data interface error codes. Error code list:

| Name | Description   |
|------|---|
| -1   | The player did not detect an available video address.   |
| -2   | Access to video data timed out.   |
| 1    | Video data loading was interrupted.<br>Possible reasons:<br>Network interrupted.<br>Browser abnormal interruption.<br>Solution:<br>Peek at the network request information in the web console to confirm if network requests are normal.<br>Restart the playback process. |



| [  |  |
|----|--|
| 2  | Possible cause: Network interrupts.  |
| 2  | Peek at the network request information in the web console to confirm if network requests are normal.<br>Restart the playback process                                |
|    | nestart the playback process.  |
|    | An error occurred during video decoding.<br>Possible cause: Abnormal video data, decoder failed to decode.   |
| 3  | Try re-transcoding and then play it again to eliminate issues introduced by the transcoding process.<br>Confirm whether the original video is normal.                |
|    | Please contact technical support and provide play parameters for troubleshooting.  |
|    | The video cannot be loaded due to unsupported format or server/network issues.<br>Possible reasons:  |
|    | Cannot access video data, CDN resource does not exist or does not return video data.<br>The current playback environment does not support this video format.         |
| 4  | Solution:<br>Peek at the network request information in the browser console to confirm if the video data request is  |
|    | normal.<br>Confirm whether the playback script for the corresponding video format has been loaded as per the   |
|    | documentation.<br>Confirm whether the current browser and page environment support the video format to be played.  |
|    | Please contact technical support and provide play parameters for troubleshooting.  |
|    | An error occurred during video decryption.<br>Possible reasons:  |
|    | The decryption key is incorrect.   |
| 5  | The request for the key API returned an exception.   |
|    | Solution:  |
|    | Confirm whether the key is correct and whether the key API returns normally.   |
|    | Please contact technical support and provide play parameters for troubleshooting.  |
|    | The VOD media data interface request timed out. When accessing media data, if the player retries 3 times without any response, this error will be thrown.            |
|    | Possible reasons:<br>The current network environment cannot connect to the media data interface, or the media data   |
| 10 | interface has been hijacked.   |
|    | Media data interface error.<br>Solution:   |
|    | Try opening the demo page we provided to see if it can play normally.  |
|    | Please contact technical support and provide play parameters for troubleshooting.  |
| 11 | The VOD media data interface did not return any data. When accessing media data, if the player retries 3 times without any data returned, this error will be thrown. |



|       | Possible reasons:<br>The current network environment cannot connect to the media data interface, or the media data<br>interface has been hijacked.<br>Media data interface error.<br>Solution:<br>Try opening the demo page we provided to see if it can play normally.<br>Please contact technical support and provide play parameters for troubleshooting.   |
|-------|--|
| 12    | The VOD media data interface returns abnormal data. When accessing media data, if the player retries 3 times and still returns data that cannot be parsed, this error will be thrown.<br>Possible reasons:<br>The current network environment cannot connect to the media data interface, or the media data interface has been hijacked.<br>The play parameters are incorrect; the media data interface cannot process them.<br>Media data interface error.<br>Solution:<br>Try opening the demo page we provided to see if it can play normally.<br>Please contact technical support and provide play parameters for troubleshooting. |
| 13    | The player did not detect video data that can be played on the current player; please transcode this video.  |
| 14    | Playing HLS under the HTML5 + hls.js pattern encounters a network error. Details can be peeked at in event.source. For a detailed introduction, please see the official hls.js documentation Network Errors.   |
| 15    | Playing HLS under the HTML5 + hls.js pattern encounters a Multimedia error. Details can be peeked at in event.source. For a detailed introduction, please see the official hls.js documentation Media Errors.  |
| 16    | Playing HLS under the HTML5 + hls.js pattern encounters a multiplexing exception. Details can be peeked at in event.source. For a detailed introduction, please see the official hls.js documentation Mux Errors.  |
| 17    | Playing HLS under the HTML5 + hls.js pattern encounters other exceptions. Details can be peeked at in event.source. For a detailed introduction, please see the official hls.js documentation Other Errors.  |
| 1013  | Player Signature is missing the contentInfo field  |
| 10008 | The media data service did not find media data corresponding to the play parameters, please confirm that the appID and fileID request parameters are correct, and that the corresponding media data has not been deleted.  |
| -2002 | Live Event Broadcasting pull stream interface returns an error in the backend (for example, the stream does not exist, authentication failure, etc.)   |
| -2006 | Live Event Broadcasting multi-resolution smooth switching interface request failed   |

# iOS

Last updated : 2024-04-11 16:11:38

## TXVodPlayer

### **VOD** player

See TXVodPlayer.

The player pulls audio/video data from the specified VOD stream URL and plays the data after decoding and local rendering.

The player has the following capabilities:

Play FLV, MP4, and HLS files by URL (general playback) or by file ID (VOD playback).

Take screenshots of the video stream.

Adjust the brightness, volume level, and playback progress using gestures.

Change the resolution manually or select a resolution automatically according to current network bandwidth.

Change the playback speed, flip the video, and use hardware decoding for acceleration.

For more information on the capabilities of the player, see Video Playback Overview.

#### **Player configuration APIs**

| API                   | Description  |  |  |
|-----------------------|--|--|--|
| config                | Configures VOD. For more information on the configuration, see TXVodPlayConfig.  |  |  |
| isAutoPlay            | Sets whether to start playback automatically after startVodPlay is called. Autoplay is on by default.  |  |  |
| token                 | Sets the token for HLS encryption. After the token is set, the player will automatically add voddrm.token.TOKEN TextureView before the filename in the URL.  |  |  |
| Іоор                  | Sets whether to loop SurfaceView .   |  |  |
| enableHWAcceleration  | Sets whether to enable hardware acceleration.  |  |  |
| setExtentOptionInfo   | Sets the player business parameters in the format of <pre>NSString *, id&gt;</pre> .   |  |  |
| setSubtitleStyle      | Set the subtitle style information, and the subtitle style can be updated after playback (only supported by the advanced version of the player).   |  |  |
| setAudioNormalization | Setting volume normalization, loudness range: -70 to 0 (LUFS). Supported from player version 11.7 of the premium version. You can set the volume normalization to preset values or custom values. The preset |  |  |



values are defined in the related classes or files, such asTXVodConstants for Android and TXVodPlayConfig.hfor iOS: Off: AUDIO\_NORMALIZATION\_OFF On: AUDIO\_NORMALIZATION\_STANDARD (standard) AUDIO\_NORMALIZATION\_LOW (low) AUDIO\_NORMALIZATION\_HIGH (high) Custom values can be set in the range of -70 to 0 LUFS, from low to high.

#### **Basic playback APIs**

| API                    | Description  |
|------------------------|--|
| startVodPlay           | Plays a video from an HTTP URL. Since v10.7, startPlay has<br>been replaced by startVodPlay, and you need to call<br>V2TXLivePremier#setLicence Or<br>TXLiveBase#setLicence to set the license in order to use the<br>playback feature (you only need to set the license once). Otherwise,<br>playback will fail (black screen). You can use a live stream publishing<br>license, UGSV license, or video playback license to activate the<br>playback feature.                   |
| startPlayDrm:          | Start a standard FairPlay DRM playback.  |
| startVodPlayWithParams | Plays a video by VOD file ID. Since v10.7, startPlayWithParams<br>has been replaced by startVodPlayWithParams, and you need<br>to call V2TXLivePremier#setLicence or<br>TXLiveBase#setLicence to set the license in order to use the<br>playback feature (you only need to set the license once). Otherwise,<br>playback will fail (black screen). You can use a live stream publishing<br>license, UGSV license, or video playback license to activate the<br>playback feature. |
| stopPlay               | Stops the audio/video stream.  |
| isPlaying              | Gets whether playback is ongoing.  |
| pause                  | Pauses playback. The player will stop pulling data and freeze on the last frame.   |
| resume                 | Resumes playback. The player will resume pulling data.   |
| seek                   | Seeks to a specified time point of the video stream (in seconds).  |
| currentPlaybackTime    | Gets the current playback time point in seconds.   |
| duration               | Gets the total video duration in seconds.  |

| playableDuration                 | Gets the playable video duration in seconds.   |  |  |
|----------------------------------|--|--|--|
| width                            | Gets the video width.  |  |  |
| height                           | Gets the video height.   |  |  |
| setStartTime                     | Sets the playback start time.  |  |  |
| setupVideoWidget:insertIndex:    | Create a Video rendering View, which hosts the display of video content.   |  |  |
| removeVideoWidget                | Remove Video rendering View .  |  |  |
| addSubtitleSource:name:mimeType: | Add external subtitles (only supported by the advanced version of the player).   |  |  |
| getSubtitleTrackInfo             | Returns the list of subtitle track information (only supported by the advanced version of the player).   |  |  |
| getAudioTrackInfo                | Returns the list of audio track information (supported only by the advanced version of the player).  |  |  |
| selectTrack:                     | Select a track (supported only in the premium version of the player).  |  |  |
| deselectTrack:                   | Deselects a track (only supported in advanced versions of the player).   |  |  |
| seekToPdtTime                    | Seek to the specified Program Date Time (PDT) point in the video<br>stream, which enables functions such as fast-forward, rewind, and<br>progress bar jumping. Currently, only HLS video format is supported.<br>This function is supported starting from version 11.6 of the player's<br>advanced edition. The parameter unit is milliseconds (ms). |  |  |

### Video APIs

| API             | Description  |  |
|-----------------|--|--|
| snapshot        | Gets the current video frame image.<br>Note: Because this operation is time-consuming, the screenshot will be called<br>back asynchronously. |  |
| setMirror       | Sets whether to flip the video image.  |  |
| setRate         | Sets the VOD playback speed. Default value: 1.0.   |  |
| bitrateIndex    | Returns the current playback bitrate index.  |  |
| setBitrateIndex | Sets the current playback bitrate index for seamless definition switch.<br>You may need to wait momentarily to switch the definition.        |  |



| supportedBitrates     | When the playback address is a master playlist, return the supported bitrates (resolutions). |
|-----------------------|--|
| setRenderMode         | Sets the image fill mode.  |
| setRenderRotation     | Sets the rotation.   |
| enterPictureInPicture | Enter the picture-in-picture function.   |
| exitPictureInPicture  | Exit the picture-in-picture function.  |

#### Audio APIs

| API                   | Description                                |
|-----------------------|--|
| setMute               | Sets whether to mute the player.           |
| setAudioPlayoutVolume | Sets the volume level. Value range: 0–100. |

#### **Event notification APIs**

| API                  | Description  |
|----------------------|--|
| delegate             | Sets player callbacks. We recommend you use vodDelegate instead.                 |
| vodDelegate          | Sets player callbacks.   |
| videoProcessDelegate | Sets the video rendering callback (supported only if hardware encoding is used). |

### **TRTC APIs**

You can use the following APIs to push the audio/video streams of the VOD player through TRTC. For more information on TRTC services, see the TRTC Overview page.

| API            | Description                       |
|----------------|-----------------------------------|
| attachTRTC     | Binds VOD to TRTC.                |
| detachTRTC     | Unbinds VOD from TRTC.            |
| publishVideo   | Starts pushing the video stream.  |
| unpublishVideo | Cancels pushing the video stream. |
| publishAudio   | Starts pushing the audio stream.  |
| unpublishAudio | Cancels pushing the audio stream. |

### Class method.

| API                       | Description   |
|---------------------------|---|
| getEncryptedPlayKey       | Get the encrypted playback key for content protection.  |
| isSupportPictureInPicture | Whether Picture-in-Picture (PiP) function is supported. |

## TXVodPlayListener

VOD callback notifications.

#### Basic SDK callback APIs

| API  | Description  |
|--|--|
| onPlayEvent:event:withParam:                       | VOD playback event notification. For more information, see the playback event list and event parameters. |
| onNetStatus:withParam:                             | Network status notification of the VOD player.   |
| onPlayer:pictureInPictureStateDidChange:withParam: | Picture-in-Picture state callback.   |
| onPlayer:pictureInPictureErrorDidOccur:withParam:  | Picture-in-Picture error state callback.   |

## TXVodPlayConfig

VOD player configuration class.

### **Basic configuration APIs**

| API                  | Description   |
|----------------------|---|
| connectRetryCount    | Sets the maximum number of player reconnection attempts.  |
| connectRetryInterval | Sets the player reconnection interval in seconds.   |
| timeout              | Sets the player connection timeout period in seconds.   |
| cacheFolderPath      | This interface has been deprecated. It is recommended to useTXPlayerGlobalSetting##setCacheFolderPath.<br>Sets the VOD cache directory, which takes effect for MP4 and HLS files. |
| maxCacheItems        | This interface has been deprecated. It is recommended to  |

|                          | useTXPlayerGlobalSetting#setMaxCacheSizeMB.<br>Sets the maximum number of cached files.  |
|--------------------------|--|
| playerType               | Sets the player type.<br>PLAYER_AVPLAYER: Use the system's built-in player.<br>PLAYER_THUMB_PLAYER: Use Tencent Cloud's self-developed player.   |
| headers                  | Sets custom HTTP headers.  |
| enableAccurateSeek       | Sets whether to enable accurate seek. Default value: true.   |
| autoRotate               | Sets whether to enable auto rotation. If the parameter is set to YES (default),<br>MP4 files will be automatically rotated.<br>You can get the rotation angle from the PLAY_EVT_CHANGE_ROTATION<br>callback. |
| smoothSwitchBitrate      | Sets whether to enable smooth switch for multi-bitrate HLS streams. Default value: false.  |
| progressInterval         | Sets the progress callback interval in ms.   |
| maxBufferSize            | Sets the maximum buffer size in MB.  |
| maxPreloadSize           | Sets the maximum preloading size in MB.  |
| firstStartPlayBufferTime | Sets the duration (ms) of video that needs to be loaded before playback starts.<br>Default value: 100 ms.  |
| nextStartPlayBufferTime  | Sets the minimum data to buffer when there is insufficient buffer data or after seeking is performed. Default value: 250 ms.   |
| overlayKey               | Sets the HLS security hardening encryption and decryption key.   |
| overlaylv                | Sets the HLS security hardening encryption and decryption IV.  |
| extInfoMap               | Sets the extended information.   |
| preferredResolution      | Sets the preferred resolution. In case of multi-bitrate HLS streams, the resolution specified by preferredResolution (width x height) will be played. This API works only if it is called before playback.   |
| enableRenderProcess      | Sets whether to enable post-rendering and post-processing (such as that used by the super-resolution plugin). They are enabled by default.   |
| playerPixelFormatType    | The video format of the video rendering object callback.   |
| keepLastFrameWhenStop    | Whether to keep the last frame when calling `stopPlay`, with a default value of NO.  |
| mediaType                | Set the media type. Optional values are:   |



| MEDIA_TYPE_AUTO, AUTO type (default value, not supported for adaptive |
|---|
| bitrate playback).  |
| MEDIA_TYPE_HLS_VOD, HLS on-demand media.                              |
| MEDIA_TYPE_HLS_LIVE, HLS live media.                                  |
| MEDIA_TYPE_HLS_VOD, MP4 and other general file on-demand media        |
| (supported since version 11.2).                                       |
| MEDIA_TYPE_DASH_VOD, DASH on-demand media (supported since version    |
| 11.2).  |
|   |

## TXPlayerGlobalSetting

Global configuration of the VOD player.

| API                | Description   |
|--------------------|---|
| setCacheFolderPath | Sets the cache directory of the playback engine. After setting, this directory will be first read and written during predownloading and player use. |
| setMaxCacheSize    | Sets the playback engine's maximum cache size in MB. The cache directory will be cleared automatically if its size exceeds the specified value.     |

## TXVodPreloadManager

Predownloading API class of the VOD player.

| API           | Description   |
|---------------|---|
| sharedManager | Gets a singleton object of TXVodPreloadManager .  |
| startPreload  | Starts predownloading. Before you call this API, make sure you have called<br>TXPlayerGlobalSetting#setCacheFolderPath and<br>TXPlayerGlobalSetting#setMaxCacheSize to set the cache directory and<br>maximum cache size. |
| stopPreload   | Stops predownloading.   |

### **TXVodPreloadManagerDelegate**

| API                | Description                   |
|--------------------|-------------------------------|
| onComplete:url:    | Download completion callback. |
| onError:url:error: | Download error callback.      |

# TXVodDownloadManager

Video download API class of the VOD player.

### **TXVodDownloadDataSource**

| API          | Description  |
|--------------|--|
| auth         | File ID information.                                   |
| quality      | Download definition, default original.                 |
| token        | If the address is encrypted, please fill in the token. |
| templateName | Definition template.                                   |
| fileId       | File ID.   |
| pSign        | Signature information.                                 |
| appld        | Application AppID. Required.                           |
| userName     | Account name.  |
| overlayKey   | HLS EXT-X-KEY encryption and decryption parameters.    |
| overlaylv    | Encryption and decryption parameter overlaylv.         |

### **TXVodDownloadMediaInfo**

| API                | Description   |
|--------------------|---|
| isDownloadFinished | Whether the download is complete.   |
| dataSource         | FileID download object (optional).  |
| url                | Download address.   |
| userName           | Account name.   |
| duration           | Duration.   |
| playableDuration   | Playable duration.  |
| size               | Get the total size of the downloaded file, in bytes, only valid for FileID download sources. Note: The total size refers to the size of the original file uploaded to the |



|                  | Tencent Cloud VOD console, and the size of the sub-streams after being converted to adaptive bitrate streams cannot be obtained temporarily.        |
|------------------|---|
| downloadSize     | Downloaded size, in bytes.  |
| segments         | Total number of segments.   |
| downloadSegments | Number of segments downloaded.  |
| progress         | Progress.   |
| playPath         | Play path, can be passed to TXVodPlayer for playback.   |
| speed            | Download speed, in bytes per second.  |
| downloadState    | Download status.  |
| isResourceBroken | Determine whether the downloaded video resource is damaged, and return YES if it is deleted after downloading, etc. (Supported since version 11.0). |

### TXVodDownloadManager

| API                      | Description  |
|--------------------------|--|
| shareInstance            | Get the TXVodDownloadManager instance object, singleton mode.                                      |
| setDownloadPath          | Set the download root directory.   |
| headers                  | Set the download HTTP headers.   |
| delegate                 | Set the download callback method, which must be set before downloading.                            |
| startDownloadUrl         | Start downloading using URL.   |
| startDownload            | Start downloading using FileID.  |
| stopDownload             | Stop downloading, and stop successfully when ITXVodDownloadListener.onDownloadStop is called back. |
| deleteDownloadFile       | Delete the downloaded file.  |
| deleteDownloadMediaInfo  | Delete the download information.   |
| getDownloadMediaInfoList | Get the download list information for all users.   |
| getDownloadMediaInfo     | Get the download information.  |
| getOverlayKeylv          | Get the HLS EXT-X-KEY. genRandomHexStringForHIs  |
|                          |  |

| genRandomHexStringForHIs | Get the encryption random number. |
|--------------------------|-----------------------------------|
| encryptHexStringHls:     | Encryption.                       |

### **TXVodDownloadDelegate**

| API                | Description   |
|--------------------|---|
| onDownloadStart    | Download started.                                       |
| onDownloadProgress | The download progress was updated.                      |
| onDownloadStop     | Download stopped.                                       |
| onDownloadFinish   | Download ended.   |
| onDownloadError    | An error occurred during download.                      |
| hlsKeyVerify       | The decryption key for downloading encrypted HLS files. |

#### **TXDownloadError**

#### Download Error Code

| Enumeration value     | Description                          |
|-----------------------|--------------------------------------|
| TXDownloadSuccess     | Download successful.                 |
| TXDownloadAuthFaild   | File ID authentication failed.       |
| TXDownloadNoFile      | No file of this definition.          |
| TXDownloadFormatError | Format not supported.                |
| TXDownloadDisconnet   | Network disconnected.                |
| TXDownloadHlsKeyError | Failed to obtain HLS decryption key. |
| TXDownloadPathError   | Failed to access download directory. |

### **TXVodDownloadMediaInfoState**

#### **Download Status**

| Enumeration value               | Description             |
|---------------------------------|-------------------------|
| TXVodDownloadMediaInfoStateInit | Download initial state. |
|                                 |                         |

| TXVodDownloadMediaInfoStateStart  | Download started.   |
|-----------------------------------|---------------------|
| TXVodDownloadMediaInfoStateStop   | Download stopped.   |
| TXVodDownloadMediaInfoStateError  | Download error.     |
| TXVodDownloadMediaInfoStateFinish | Download completed. |

### **TXVodQuality**

The resolution of the downloaded video.

Explanation:

The constants TXVodQuality240P ~ TXVodQuality1080p are newly added in version 11.0.

| Enumeration value | Description               |
|-------------------|---------------------------|
| TXVodQualityOD    | Original.                 |
| TXVodQualityFLU   | Smooth.                   |
| TXVodQualitySD    | Standard Definition.      |
| TXVodQualityHD    | High Definition.          |
| TXVodQualityFHD   | Full High Definition.     |
| TXVodQuality2K    | 2К.                       |
| TXVodQuality4K    | 4K.                       |
| TXVodQuality240P  | Smooth 240P.              |
| TXVodQuality360P  | Smooth 360P.              |
| TXVodQuality480P  | Standard Definition 480P. |
| TXVodQuality540P  | Standard Definition 540P. |
| TXVodQuality720P  | High Definition 720P.     |
| TXVodQuality1080p | High Definition 1080P.    |

### TXPlayerAuthParams

Play encrypted video configuration by fileId.

API

Description

| appld   | App appld.                        |
|---------|-----------------------------------|
| fileId  | File id.                          |
| timeout | Encrypted link timeout timestamp. |
| exper   | Try the duration.                 |
| us      | Uniquely identifies the request.  |
| sign    | Anti-leech signature.             |
| https   | Whether to use https requests.    |

### **TXBitrateItem**

#### HLS multi-bit rate information.

| API     | Description                           |
|---------|---------------------------------------|
| index   | The sequence number in the m3u8 file. |
| width   | The video width of this stream.       |
| height  | The video height of this stream.      |
| bitrate | The video bitrate of this stream.     |

## TXImageSprite

Sprite analysis tool.

| API          | Description                 |
|--------------|-----------------------------|
| setVTTUrl    | Set the sprite map address. |
| getThumbnail | Get a thumbnail.            |

## TXPlayerDrmBuilder

On-demand Drm builder.

| API            | Description               |
|----------------|---------------------------|
| certificateUrl | Certificate provider URL. |
| keyLicenseUrl  | Decrypt the key URL.      |
| playUrl        | play link.                |

## Error Codes

### Normal events

| Code | Event Definition           | Description   |
|------|----------------------------|---|
| 2004 | PLAY_EVT_PLAY_BEGIN        | Video playback started, and the loading icon animation (if any) ended.  |
| 2005 | PLAY_EVT_PLAY_PROGRESS     | The video playback progress (including the current playback progress, the loaded duration, and the total video duration). |
| 2007 | PLAY_EVT_PLAY_LOADING      | The video is being loaded. The LOADING_END event will be reported if video playback resumes.                              |
| 2014 | PLAY_EVT_VOD_LOADING_END   | Video loading ended, and video playback resumed.  |
| 2006 | PLAY_EVT_PLAY_END          | Video playback ended.   |
| 2013 | PLAY_EVT_VOD_PLAY_PREPARED | The player is ready.  |
| 2003 | PLAY_EVT_RCV_FIRST_I_FRAME | The network received the first renderable video data packet (IDR).  |
| 2009 | PLAY_EVT_CHANGE_RESOLUTION | The video resolution changed.   |
| 2011 | PLAY_EVT_CHANGE_ROTATION   | The MP4 video was rotated.  |

### Warnings

| Code  | Event Definition        | Description   |
|-------|-------------------------|---|
| -2301 | PLAY_ERR_NET_DISCONNECT | The network was disconnected and could not be<br>reconnected after multiple retries. You can<br>restart the player to perform more connection<br>retries. |

| -2305 | PLAY_ERR_HLS_KEY                  | Failed to get the HLS decryption key.   |
|-------|-----------------------------------|---|
| 2101  | PLAY_WARNING_VIDEO_DECODE_FAIL    | Failed to decode the current video frame.   |
| 2102  | PLAY_WARNING_AUDIO_DECODE_FAIL    | Failed to decode the current audio frame.   |
| 2103  | PLAY_WARNING_RECONNECT            | The player was disconnected and is trying to reconnect. The PLAY_ERR_NET_DISCONNECT event will be thrown after three failed attempts. |
| 2106  | PLAY_WARNING_HW_ACCELERATION_FAIL | Failed to start the hardware decoder, and the software decoder was used instead.  |
| -2304 | PLAY_ERR_HEVC_DECODE_FAIL         | H.265 decoding failed.  |
| -2303 | PLAY_ERR_FILE_NOT_FOUND           | The file to be played back does not exist.  |

# Player SDK constants.

### Event code and error code

| Enumeration value                  | Meaning   |
|------------------------------------|---|
| VOD_PLAY_EVT_RCV_FIRST_I_FRAME     | Playback event: The first video frame was successfully received.  |
| VOD_PLAY_EVT_RCV_FIRST_AUDIO_FRAME | Play event: The first audio frame was successfully received.  |
| VOD_PLAY_EVT_PLAY_BEGIN            | Play event: Play has started.   |
| VOD_PLAY_EVT_PLAY_PROGRESS         | Play event: Play progress update, dedicated to VodPlayer.   |
| VOD_PLAY_EVT_PLAY_END              | Play event: Play has ended.   |
| VOD_PLAY_EVT_PLAY_LOADING          | Play event: data buffering.   |
| VOD_PLAY_EVT_START_VIDEO_DECODER   | Playback event: The video decoder has started.  |
| VOD_PLAY_EVT_CHANGE_RESOLUTION     | Playback event: Video resolution changed.   |
| VOD_PLAY_EVT_GET_PLAYINFO_SUCC     | Play event: the information about successfully obtaining the VOD file, dedicated to the VOD player (VodPlayer). |

| VOD_PLAY_EVT_CHANGE_ROTATION             | Play event: The rotation angle of the MP4 video changes, dedicated to VodPlayer.   |
|--|--|
| VOD_PLAY_EVT_VOD_PLAY_PREPARED           | Play event: Video loading is complete, dedicated to VodPlayer.   |
| VOD_PLAY_EVT_VOD_LOADING_END             | Play event: Video buffering ends, dedicated to VodPlayer.  |
| VOD_PLAY_EVT_STREAM_SWITCH_SUCC          | Playback event: Stream switching (switching<br>between video streams with different resolutions)<br>has been successfully completed. |
| VOD_PLAY_EVT_VOD_PLAY_TCP_CONNECT_SUCC   | The TCP connection was successful.   |
| VOD_PLAY_EVT_VOD_PLAY_FIRST_VIDEO_PACKET | The first frame of data is received.   |
| VOD_PLAY_EVT_VOD_PLAY_SEEK_COMPLETE      | Video playback Seek complete.  |
| VOD_PLAY_EVT_AUDIO_SESSION_INTERRUPT     | Playback event: Audio Session is interrupted by other apps (applicable to iOS platform only).  |
| VOD_PLAY_ERR_NET_DISCONNECT              | Streaming Error: The network connection was disconnected (after three retries and failed to reconnect).                              |
| VOD_PLAY_ERR_FILE_NOT_FOUND              | On-demand error: The playback file does not exist.   |
| VOD_PLAY_ERR_HLS_KEY                     | On-demand error: Failed to obtain HLS decoding KEY.  |
| VOD_PLAY_ERR_GET_PLAYINFO_FAIL           | On-demand error: Failed to obtain the file information of the on-demand file.  |

### Picture-in-picture error type

| Enumeration value                               | Meaning   |
|---|---|
| TX_VOD_PLAYER_PIP_ERROR_TYPE_NONE               | No errors.  |
| TX_VOD_PLAYER_PIP_ERROR_TYPE_DEVICE_NOT_SUPPORT | The device or system version does<br>not support (iPad iOS9+ only supports<br>PIP). |
| TX_VOD_PLAYER_PIP_ERROR_TYPE_PLAYER_NOT_SUPPORT | Player does not support.  |
| TX_VOD_PLAYER_PIP_ERROR_TYPE_VIDEO_NOT_SUPPORT  | Video not supported.  |

| TX_VOD_PLAYER_PIP_ERROR_TYPE_PIP_IS_NOT_POSSIBLE | PIP controller not available.        |
|--|--------------------------------------|
| TX_VOD_PLAYER_PIP_ERROR_TYPE_ERROR_FROM_SYSTEM   | The PIP controller reports an error. |
| TX_VOD_PLAYER_PIP_ERROR_TYPE_PLAYER_NOT_EXIST    | The player object does not exist.    |
| TX_VOD_PLAYER_PIP_ERROR_TYPE_PIP_IS_RUNNING      | The PIP function is already running. |
| TX_VOD_PLAYER_PIP_ERROR_TYPE_PIP_NOT_RUNNING     | PIP function is not started.         |

### Picture-in-picture controller status

| Enumeration value                  | Meaning                                 |
|------------------------------------|---|
| TX_VOD_PLAYER_PIP_STATE_UNDEFINED  | Not Set.                                |
| TX_VOD_PLAYER_PIP_STATE_WILL_START | Picture-in-picture is about to start.   |
| TX_VOD_PLAYER_PIP_STATE_DID_START  | Picture-in-picture has started.         |
| TX_VOD_PLAYER_PIP_STATE_WILL_STOP  | Picture-in-picture is coming to an end. |
| TX_VOD_PLAYER_PIP_STATE_DID_STOP   | Picture-in-picture is over.             |
| TX_VOD_PLAYER_PIP_STATE_RESTORE_UI | reset UI.                               |

# Android

Last updated : 2024-04-11 16:11:38

## TXVodPlayer

### **VOD** player

#### See TXVodPlayer.

The player pulls audio/video data from the specified VOD stream URL and plays the data after decoding and local rendering.

The player has the following capabilities:

Play FLV, MP4, and HLS files by URL (general playback) or by file ID (VOD playback).

Take screenshots of the video stream.

Adjust the brightness, volume level, and playback progress using gestures.

Change the resolution manually or select a resolution automatically according to current network bandwidth.

Change the playback speed, flip the video, and use hardware decoding for acceleration.

For more information on the capabilities of the player, see Video Playback Overview.

#### **Player configuration APIs**

| API                   | Description   |  |
|-----------------------|---|--|
| setConfig             | Sets the player configuration information. For more information on the configuration, see TXVodPlayConfig.  |  |
| setPlayerView         | Sets the player's rendering view TXCloudVideoView .   |  |
| setPlayerView         | Sets the player's rendering view TextureView .  |  |
| setSurface            | Sets the player's rendering view SurfaceView .  |  |
| setStringOption       | Sets the player business parameters in the format of <string, object=""> .</string,>  |  |
| setSubtitleStyle      | Sets the subtitle style information and can update the subtitle style after playback (only supported in the advanced version of the player).  |  |
| setSubtitleView       | Sets the subtitle software decoding target object (only supported in the advanced version of the player).   |  |
| setAudioNormalization | Setting volume normalization, loudness range: -70 to 0 (LUFS). Supported from player version 11.7 of the premium version.<br>You can set the volume normalization to preset values or custom values. The preset |  |



| values are defined in the related classes or files, such asTXVodConstants for Android |
|---|
| and TXVodPlayConfig.hfor iOS:   |
| Off: AUDIO_NORMALIZATION_OFF  |
| On: AUDIO_NORMALIZATION_STANDARD (standard)   |
| AUDIO_NORMALIZATION_LOW (low)   |
| AUDIO_NORMALIZATION_HIGH (high)   |
| Custom values can be set in the range of -70 to 0 LUFS, from low to high.             |

### **Basic playback APIs**

| API                    | Description  |
|------------------------|--|
| startVodPlay           | Plays a video from an HTTP URL. Since v10.7, startPlay has been replaced<br>by startVodPlay, and you need to call V2TXLivePremier#setLicence<br>or TXLiveBase#setLicence to set the license in order to use the playback<br>feature (you only need to set the license once). Otherwise, playback will fail (black<br>screen). You can use a live stream publishing license, UGSV license, or video<br>playback license to activate the playback feature. |
| startVodPlay           | Plays a video by VOD file ID. Since v10.7, startPlay has been replaced by startVodPlay, and you need to call V2TXLivePremier#setLicence or TXLiveBase#setLicence to set the license in order to use the playback feature (you only need to set the license once). Otherwise, playback will fail (black screen). You can use a live stream publishing license, UGSV license, or video playback license to activate the playback feature.                  |
| stopPlay               | Stops playback.  |
| isPlaying              | Gets whether playback is ongoing.  |
| pause                  | Pauses playback. The player will stop pulling data and freeze on the last frame.   |
| resume                 | Resumes playback. The player will resume pulling data.   |
| seek                   | Seeks to a specified time point of the video stream (in seconds).  |
| seek                   | Seeks to a specified time point of the video stream (in ms).   |
| getCurrentPlaybackTime | Gets the current playback time point in seconds.   |
| getBufferDuration      | Gets the total buffer duration in seconds.   |
| getDuration            | Gets the total video duration in seconds.  |
| getPlayableDuration    | Gets the playable video duration in seconds.   |
| getWidth               | Gets the video width.  |



| getHeight            | Gets the video height.   |  |
|----------------------|--|--|
| setAutoPlay          | Sets whether to start playback automatically after startVodPlay is called.<br>Auto-play is on by default.  |  |
| setStartTime         | Sets the playback start time.  |  |
| setToken             | Sets the token for HLS encryption.   |  |
| setLoop              | Sets whether to loop the video.  |  |
| isLoop               | Gets whether playback is looped.   |  |
| addSubtitleSource    | Adds external subtitles (only supported in the advanced version of the player).  |  |
| getSubtitleTrackInfo | Returns a list of subtitle track information (only supported in the advanced version of the player).   |  |
| getAudioTrackInfo    | Returns a list of audio track information (only supported in the advanced version of the player).  |  |
| selectTrack          | Selects a track (only supported in the advanced version of the player).  |  |
| deselectTrack        | Deselects a track (only supported in the advanced version of the player).  |  |
| seekToPdtTime        | Seek to the specified Program Date Time (PDT) point in the video stream, which<br>enables functions such as fast-forward, rewind, and progress bar jumping.<br>Currently, only HLS video format is supported. This function is supported starting<br>from version 11.6 of the player's advanced edition. The parameter unit is<br>milliseconds (ms). |  |

### Video APIs

| API                  | Description  |
|----------------------|--|
| enableHardwareDecode | Enables/Disables video hardware decoding.  |
| snapshot             | Gets the current video frame image.<br>Note: Because this operation is time-consuming, the screenshot will be<br>called back asynchronously. |
| setMirror            | Sets whether to flip the video image horizontally.   |
| setRate              | Sets the VOD playback speed. Default value: 1.0.   |
| getBitrateIndex      | Returns the current playback bitrate index.  |
| setBitrateIndex      | Sets the current playback bitrate index for seamless definition switch. You may need   |



|                   | to wait momentarily to switch the definition. |
|-------------------|---|
| setRenderMode     | Sets the image fill mode.                     |
| setRenderRotation | Sets the rotation.                            |

#### **Audio APIs**

| API                   | Description  |
|-----------------------|--|
| setMute               | Sets whether to mute the player.   |
| setAudioPlayoutVolume | Sets the volume level. Value range: 0–100.   |
| setRequestAudioFocus  | Sets whether to get the audio focus automatically. The audio focus is gotten automatically by default. |

#### **Event notification APIs**

| API             | Description   |
|-----------------|---|
| setPlayListener | Sets player callbacks. This API has been deprecated. Please use setVodListener instead. |
| setVodListener  | Sets player callbacks.  |

### **TRTC APIs**

You can use the following APIs to push the audio/video streams of the VOD player through TRTC. For more information on TRTC services, see the TRTC Overview page.

| API            | Description                       |
|----------------|-----------------------------------|
| attachTRTC     | Binds VOD to TRTC.                |
| detachTRTC     | Unbinds VOD from TRTC.            |
| publishVideo   | Starts pushing the video stream.  |
| unpublishVideo | Cancels pushing the video stream. |
| publishAudio   | Starts pushing the audio stream.  |
| unpublishAudio | Cancels pushing the audio stream. |

## ITXVodPlayListener

VOD callback notifications.

### **Basic SDK callback APIs**

| API         | Description  |
|-------------|--|
| onPlayEvent | VOD playback event notification. For more information, see the playback event list and event parameters. |
| onNetStatus | Network status notification of the VOD player.   |

## TXVodPlayConfig

VOD player configuration class.

### **Basic configuration APIs**

| API                     | Description   |
|-------------------------|---|
| setConnectRetryCount    | Sets the maximum number of player reconnection attempts.  |
| setConnectRetryInterval | Sets the player reconnection interval in seconds.   |
| setTimeout              | Sets the player connection timeout period in seconds.   |
| setCacheFolderPath      | Sets the VOD cache directory, which takes effect for MP4 and HLS files.   |
| setMaxCacheItems        | Sets the maximum number of cached files. This API has been deprecated. Use<br>TXPlayerGlobalSetting#setMaxCacheSize for global configuration.   |
| setPlayerType           | Sets the player type.<br>TXVodConstants#PLAYER_SYSTEM_MEDIA_PLAYER: Use the system's built-in<br>player.<br>TXVodConstants#PLAYER_THUMB_PLAYER: Use Tencent Cloud's self-<br>developed player.      |
| setHeaders              | Sets custom HTTP headers.   |
| setEnableAccurateSeek   | Sets whether to enable accurate seek. Default value: true.  |
| setAutoRotate           | Sets whether to enable auto rotation. If the parameter is set to YES (default), MP4 files will be automatically rotated. You can get the rotation angle from the PLAY_EVT_CHANGE_ROTATION callback. |



| setSmoothSwitchBitrate | Sets whether to enable smooth switching for multi-bitrate HLS streams. Default value: false.   |  |
|------------------------|--|--|
| setCacheMp4ExtName     | Sets the extension for cached MP4 files.   |  |
| setProgressInterval    | Sets the progress callback interval in ms.   |  |
| setMaxBufferSize       | Sets the maximum buffer size in MB.  |  |
| setMaxPreloadSize      | Set the maximum buffer size for preloading, in MB.   |  |
| setExtInfo             | Set extended information.  |  |
| setPreferredResolution | When playing HLS with multiple streams, select the optimal stream for playback based on the set preferredResolution, which is the product of the width and height (width * height). This setting takes effect before playback.   |  |
| setOverlayKey          | Set the encryption and decryption key for HLS security reinforcement.  |  |
| setOverlaylv           | Set the encryption and decryption IV for HLS security reinforcement.   |  |
| setEnableRenderProcess | Enable or disable loading and rendering post-processing services.  |  |
| setMediaType           | Set the media type, with AUTO as the default value. Optional values are:<br>TXVodConstants#MEDIA_TYPE_AUTO, AUTO type (default value, adaptive bit<br>rate playback is not supported for now);<br>TXVodConstants#MEDIA_TYPE_HLS_VOD, HLS VOD media;<br>TXVodConstants#MEDIA_TYPE_HLS_LIVE, HLS live media;<br>TXVodConstants#MEDIA_TYPE_HLS_VOD, MP4 and other common file VOD<br>media (supported starting from version 11.2);<br>TXVodConstants#MEDIA_TYPE_DASH_VOD, DASH VOD media (supported<br>starting from version 11.2); |  |

## TXPlayerGlobalSetting

Global configuration of the VOD player.

| API                | Description   |
|--------------------|---|
| setCacheFolderPath | Sets the cache directory of the playback engine. After setting, this directory will be first read and written during downloading, predownloading, and player use. |
| setMaxCacheSize    | Sets the playback engine's maximum cache size in MB. The cache directory will be cleared automatically if its size exceeds the specified value.                   |
| setDrmProvisionEnv | Set the Drm certificate provider environment (supported starting from version 11.2).  |

Optional values are: TXPlayerGlobalSetting.DrmProvisionEnv#DRM\_PROVISION\_ENV\_COM, which represents the use of the Google COM domain name certificate provider; TXPlayerGlobalSetting.DrmProvisionEnv#DRM\_PROVISION\_ENV\_CN, which represents the use of the Google CN domain name certificate provider;

### TXVodPreloadManager

Predownloading API class of the VOD player.

| API          | Description   |
|--------------|---|
| getInstance  | Gets a singleton object of TXVodPreloadManager .  |
| startPreload | Starts predownloading. Before you call this API, make sure you have called<br>TXPlayerGlobalSetting#setCacheFolderPath and<br>TXPlayerGlobalSetting#setMaxCacheSize to set the cache directory and maximum<br>cache size. |
| stopPreload  | Stops predownloading.   |

## ITXVodPreloadListener

#### Video pre-download callback interface.

| API        | Description                              |
|------------|--|
| onComplete | Video pre-download is complete.          |
| onError    | Video pre-download encountered an error. |

## TXVodDownloadManager

Video download API class of the VOD player. Currently, only non-nested M3U8 videos can be downloaded.

SimpleAES-encrypted videos will be encrypted again with Tencent Cloud's private encryption algorithm to improve the security.

| API         | Description                |                        |
|-------------|----------------------------|------------------------|
| getInstance | Gets a singleton object of | TXVodDownloadManager . |
|             |                            |                        |



| setHeaders               | Sets download HTTP headers.   |
|--------------------------|---|
| setListener              | Sets the download callback. This API must be called before the download.                                      |
| startDownloadUrl         | Starts downloading the video at the specified URL.  |
| startDownload            | Starts downloading the video of the specified fileid .  |
| stopDownload             | Stops the download. If ITXVodDownloadListener.onDownloadStop is called back, the download stops successfully. |
| deleteDownloadMediaInfo  | Deletes the download information.   |
| getDownloadMediaInfoList | Gets the download lists of all users.   |
| getDownloadMediaInfo     | Gets the download information.  |

## ITXVodDownloadListener

VOD download callbacks.

| API                | Description   |
|--------------------|---|
| onDownloadStart    | Download started.   |
| onDownloadProgress | The download progress was updated.  |
| onDownloadStop     | Download stopped.   |
| onDownloadFinish   | Download ended.   |
| onDownloadError    | An error occurred during download.  |
| hlsKeyVerify       | When downloading HLS, if an encrypted file is encountered, the decryption key is passed to the external party for verification. |

## TXVodDownloadDataSource

Tencent Cloud download source ( fileid ) APIs. You need to pass in the source information when starting download.

| API                     | Description  |
|-------------------------|--|
| TXVodDownloadDataSource | Builds a function to pass in parameters such as appid, fileid, |

|             | quality, psign, and username.                                     |
|-------------|---|
| getAppId    | Gets the appid that is passed in.                                 |
| getFileId   | Gets the fileid that is passed in.                                |
| getPSign    | Gets the psign that is passed in.                                 |
| getQuality  | Gets the quality that is passed in.                               |
| getUserName | Gets the userName that is passed in, which is default by default. |

#### Video quality constants

#### Description:

TXVodQuality240P ~ TXVodQuality1080p constants were added in version 11.0.

| code | Constant Definition | Description                |
|------|---------------------|----------------------------|
| 0    | QUALITY_OD          | Original quality           |
| 1    | QUALITY_FLU         | Smooth quality             |
| 2    | QUALITY_SD          | Standard definition        |
| 3    | QUALITY_HD          | High definition            |
| 4    | QUALITY_FHD         | Full high definition       |
| 5    | QUALITY_2K          | 2К                         |
| 6    | QUALITY_4K          | 4K                         |
| 1000 | QUALITY_UNK         | Undefined quality          |
| 240  | TXVodQuality240P    | Smooth 240P                |
| 360  | TXVodQuality360P    | Smooth 360P                |
| 480  | TXVodQuality480P    | Standard definition 480P   |
| 540  | TXVodQuality540P    | Standard definition 540P   |
| 720  | TXVodQuality720P    | High definition 720P       |
| 1080 | TXVodQuality1080p   | Full high definition 1080P |

## TXVodDownloadMediaInfo

Tencent Cloud download information APIs, which can be used to get the download progress, playback URL, and more.

| API                 | Description  |
|---------------------|--|
| getDataSource       | Gets the source specified by the fileid passed in for download.  |
| getDuration         | Gets the total duration of the video.  |
| getPlayableDuration | Gets the downloaded (playable) duration of the video.  |
| getSize             | Gets the total size of the file being downloaded. This API takes effect only for download by fileid .  |
| getDownloadSize     | Gets the size of the data already downloaded. This API takes effect only for download by fileid .  |
| getProgress         | Gets the current download progress.  |
| getPlayPath         | Gets the current playback path, which can be passed in to TXVodPlayer for playback.  |
| getDownloadState    | Gets the download status.  |
| isDownloadFinished  | Gets whether the download is completed.  |
| getSpeed            | Get download speed, in KBytes/second. (Supported starting from version 10.9)   |
| isResourceBroken    | Determine whether the downloaded video resource is damaged. If the downloaded file is deleted, etc., true will be returned. (Supported starting from version 11.0) |

## **Static Properties**

| Code | Property Definition | Description               |
|------|---------------------|---------------------------|
| 0    | STATE_INIT          | Initial state of download |
| 1    | STATE_START         | Download started          |
| 2    | STATE_STOP          | Download stopped          |
| 3    | STATE_ERROR         | Download error            |
|      |                     |                           |

| 4 STATE_FINISH Download completed |  |
|-----------------------------------|--|
|-----------------------------------|--|

# TXPlayerAuthBuilder

Configuration for playing encrypted videos using fileId.

| API        | Description                           |
|------------|---------------------------------------|
| setAppId   | Application appld.                    |
| setFileId  | File id.                              |
| setTimeout | Timeout for encrypted links.          |
| setExper   | Trial duration.                       |
| setUs      | Unique identifier for the request.    |
| setSign    | Anti-leech signature.                 |
| setHttps   | Whether to use HTTPS for the request. |

### **TXBitrateItem**

#### Video bitrate information.

| API       | Description  |
|-----------|--|
| index     | Sequence number in the m3u8 file.                      |
| width     | Video width of this stream.                            |
| height    | Video height of this stream.                           |
| bitrate   | Video bitrate of this stream.                          |
| compareTo | Compare whether the bitrates of two streams are equal. |

### **TXImageSprite**

Sprite image parsing class.

API

Description

| TXImageSprite         | Constructor.  |
|-----------------------|---|
| setVTTUrlAndImageUrls | Set the sprite image address.                         |
| getThumbnail          | Get the thumbnail.                                    |
| release               | Call this method when finished to avoid memory leaks. |

# TXPlayerDrmBuilder

#### DRM playback information.

| API                     | Description                                    |
|-------------------------|--|
| TXPlayerDrmBuilder      | Construct the DRM playback information object. |
| setDeviceCertificateUrl | Set the certificate provider URL.              |
| setKeyLicenseUrl        | Set the decryption key URL.                    |
| setPlayUrl              | Set the URL for playing media.                 |

## **TXPlayInfoParams**

Video playback parameter information using fileId.

| API              | Description                                       |
|------------------|---|
| TXPlayInfoParams | Constructor.                                      |
| getAppId         | Get the application ID.                           |
| getFileId        | Get the file ID.                                  |
| getPSign         | Get the Tencent Cloud video encryption signature. |

## Error Codes

### Normal events

| Code | Event Definition | Description |
|------|------------------|-------------|
|      |                  |             |

| 2004 | PLAY_EVT_PLAY_BEGIN        | Video playback started, and the loading icon animation (if any) ended.  |
|------|----------------------------|---|
| 2005 | PLAY_EVT_PLAY_PROGRESS     | The video playback progress (including the current playback progress, the loaded duration, and the total video duration). |
| 2007 | PLAY_EVT_PLAY_LOADING      | The video is being loaded. The LOADING_END event will be reported if video playback resumes.                              |
| 2014 | PLAY_EVT_VOD_LOADING_END   | Video loading ended, and video playback resumed.  |
| 2006 | PLAY_EVT_PLAY_END          | Video playback ended.   |
| 2013 | PLAY_EVT_VOD_PLAY_PREPARED | The player is ready.  |
| 2003 | PLAY_EVT_RCV_FIRST_I_FRAME | The first renderable video data packet (IDR) was received.  |
| 2009 | PLAY_EVT_CHANGE_RESOLUTION | The video resolution changed.   |
| 2011 | PLAY_EVT_CHANGE_ROTATION   | The MP4 video was rotated.  |

### Warnings

| Code  | Event Definition                  | Description   |
|-------|-----------------------------------|---|
| -2301 | PLAY_ERR_NET_DISCONNECT           | The network was disconnected and could not be<br>reconnected after multiple retries. You can<br>restart the player to perform more connection<br>retries. |
| -2305 | PLAY_ERR_HLS_KEY                  | Failed to get the HLS decryption key.   |
| 2101  | PLAY_WARNING_VIDEO_DECODE_FAIL    | Failed to decode the current video frame.   |
| 2102  | PLAY_WARNING_AUDIO_DECODE_FAIL    | Failed to decode the current audio frame.   |
| 2103  | PLAY_WARNING_RECONNECT            | The player was disconnected and is trying to reconnect. The<br>PLAY_ERR_NET_DISCONNECT event will be thrown after three failed attempts.                  |
| 2106  | PLAY_WARNING_HW_ACCELERATION_FAIL | Failed to start the hardware decoder, and the software decoder was used instead.  |
| -2304 | PLAY_ERR_HEVC_DECODE_FAIL         | H.265 decoding failed.  |
|       |                                   |   |

| -2303 | PLAY_ERR_FILE_NOT_FOUND | The file to be played back does not exist. |
|-------|-------------------------|--|

# Constants in the player SDK

The following constants have been available since version 10.0 via TXVodConstants :

#### Image tiling modes

| code | Event definitions             | Description                          |
|------|-------------------------------|--------------------------------------|
| 0    | RENDER_MODE_FULL_FILL_SCREEN  | Full screen display of video.        |
| 1    | RENDER_MODE_ADJUST_RESOLUTION | Adaptive display of video on screen. |

#### Image rendering angles

| code | Event Definition          | Description                      |
|------|---------------------------|----------------------------------|
| 0    | RENDER_ROTATION_PORTRAIT  | Normal portrait.                 |
| 270  | RENDER_ROTATION_LANDSCAPE | Rotated 90 degrees to the right. |

### List of playback events

| code | Event Definition                 | Description   |
|------|----------------------------------|---|
| 2003 | VOD_PLAY_EVT_RCV_FIRST_I_FRAME   | The network has received the first video data packet (IDR). |
| 2004 | VOD_PLAY_EVT_PLAY_BEGIN          | Video playback has started.                                 |
| 2005 | VOD_PLAY_EVT_PLAY_PROGRESS       | Video playback progress.                                    |
| 2006 | VOD_PLAY_EVT_PLAY_END            | Video playback has ended.                                   |
| 2007 | VOD_PLAY_EVT_PLAY_LOADING        | Video playback is loading.                                  |
| 2008 | VOD_PLAY_EVT_START_VIDEO_DECODER | The decoder has started.                                    |
| 2009 | VOD_PLAY_EVT_CHANGE_RESOLUTION   | Video resolution has changed.                               |
| 2010 | VOD_PLAY_EVT_GET_PLAYINFO_SUCC   | Successfully obtained information about the VOD file.       |
| 2011 | VOD_PLAY_EVT_CHANGE_ROTATION     | Video rotation information.                                 |
# 🔗 Tencent Cloud

| 2013  | VOD_PLAY_EVT_VOD_PLAY_PREPARED        | Video has finished loading.  |
|-------|---------------------------------------|--|
| 2014  | VOD_PLAY_EVT_VOD_LOADING_END          | Loading has ended.   |
| 2026  | VOD_PLAY_EVT_RCV_FIRST_AUDIO_FRAME    | The first audio frame has been played.   |
| 2103  | VOD_PLAY_WARNING_RECONNECT            | Network disconnected, automatic reconnection has been started.   |
| -2301 | VOD_PLAY_ERR_NET_DISCONNECT           | Network disconnected, and multiple reconnection attempts have failed.  |
| -2303 | VOD_PLAY_ERR_FILE_NOT_FOUND           | File not found.  |
| -2304 | VOD_PLAY_ERR_HEVC_DECODE_FAIL         | H.265 decoding failed.   |
| -2305 | VOD_PLAY_ERR_HLS_KEY                  | Failed to obtain the HLS decryption key.   |
| -2306 | VOD_PLAY_ERR_GET_PLAYINFO_FAIL        | Failed to obtain information about the VOD file.   |
| 2106  | VOD_PLAY_WARNING_HW_ACCELERATION_FAIL | Hardware acceleration failed and software decoding is being used.  |
| -5    | VOD_PLAY_ERR_INVALID_LICENSE          | Invalid license, playback failed. Note:<br>Before calling startVodPlay, you need to<br>set the license using<br>TXLiveBase#setLicence to play<br>successfully. Otherwise, playback will fail<br>(black screen). Set the license only once<br>globally. Live streaming licenses, short<br>video licenses, and video playback<br>licenses can be used. If you have not<br>obtained the above licenses, you can<br>apply for a free trial license to play<br>normally. The official license needs to be<br>purchased. |

# Playback event parameters

| Event Definition | Description                   |  |
|------------------|-------------------------------|--|
| EVT_UTC_TIME     | UTC time.                     |  |
| EVT_TIME         | Time when the event occurred. |  |
| EVT_DESCRIPTION  | Event description.            |  |

| EVT_PARAM1                   | Event parameter 1.   |
|------------------------------|--|
| EVT_PARAM2                   | Event parameter 2.   |
| EVT_PLAY_COVER_URL           | Video cover URL.   |
| EVT_PLAY_URL                 | Video URL.   |
| EVT_PLAY_NAME                | Video name.  |
| EVT_PLAY_DESCRIPTION         | Video description.   |
| EVT_PLAY_PROGRESS_MS         | Playback progress (in milliseconds).                                 |
| EVT_PLAY_DURATION_MS         | Total playback time (in milliseconds).                               |
| EVT_PLAY_PROGRESS            | Playback progress.   |
| EVT_PLAY_DURATION            | Total playback time.   |
| EVT_PLAYABLE_DURATION_MS     | Playable duration of VOD (in milliseconds).                          |
| EVT_PLAYABLE_RATE            | Playback rate.   |
| EVT_PLAYABLE_DURATION        | Playable duration of VOD.  |
| EVT_IMAGESPRIT_WEBVTTURL     | URL for downloading the webVTT description file of the sprite image. |
| EVT_IMAGESPRIT_IMAGEURL_LIST | URL list for downloading the sprite image.                           |
| EVT_DRM_TYPE                 | Encryption type.   |
| EVT_CODEC_TYPE               | Video encoding type.   |
| EVT_KEY_FRAME_CONTENT_LIST   | Video keyframe description information.                              |
| EVT_KEY_FRAME_TIME_LIST      | Keyframe time.   |

# Playback network status notification parameters

| Event Definition        | Description   |
|-------------------------|---------------|
| NET_STATUS_CPU_USAGE    | CPU usage.    |
| NET_STATUS_VIDEO_WIDTH  | Video width.  |
| NET_STATUS_VIDEO_HEIGHT | Video height. |
|                         |               |

| NET_STATUS_VIDEO_FPS     | Current video frame rate.                |
|--------------------------|--|
| NET_STATUS_VIDEO_GOP     | Current video GOP.                       |
| NET_STATUS_VIDEO_BITRATE | Video bitrate.                           |
| NET_STATUS_AUDIO_BITRATE | Audio bitrate.                           |
| NET_STATUS_NET_SPEED     | Network speed.                           |
| NET_STATUS_AUDIO_CACHE   | Number of audio frames.                  |
| NET_STATUS_VIDEO_CACHE   | Number of video frames.                  |
| NET_STATUS_AUDIO_INFO    | Audio information of the current stream. |
| NET_STATUS_NET_JITTER    | Network jitter.                          |
| NET_STATUS_SERVER_IP     | Server IP address connected to.          |
| NET_STATUS_VIDEO_DPS     | Current decoder output frame rate.       |
| NET_STATUS_QUALITY_LEVEL | Network quality level.                   |

# Player media types

| Code | Event Definition    | Description                                  |
|------|---------------------|--|
| 0    | MEDIA_TYPE_AUTO     | Auto type.                                   |
| 1    | MEDIA_TYPE_HLS_VOD  | Adaptive bitrate playback of HLS VOD media.  |
| 2    | MEDIA_TYPE_HLS_LIVE | Adaptive bitrate playback of HLS live media. |

# Uncategorized variables

| Code | Event Definition | Description                        |
|------|------------------|------------------------------------|
| -1   | INDEX_AUTO       | Adaptive bitrate index identifier. |

# Flutter

Last updated : 2024-04-26 11:09:31



# setGlobalLicense

#### Description

This API is used to set the license.

After you apply for and get a license, you can use the following API to initialize it. We recommend you call this API during application start. Video playback will fail if the license is not set.





static Future<void> setGlobalLicense(String licenceUrl, String licenceKey) async;

# Parameter description

| Parameter  | Туре   | Description     |
|------------|--------|-----------------|
| licenceUrl | String | The license URL |
| licenceKey | String | The license key |

### Return values

Unlimited

# createVodPlayer

#### Description

This API is used to create a VOD player instance at the native layer. If you use TXVodPlayerController, as it already integrates an instance, you don't need to create another.

#### API



static Future<int?> createVodPlayer() async;

# **Parameter description**

Unlimited

#### **Return values**

| Returned Value | Туре | Description   |
|----------------|------|---------------|
| playerId       | int  | The player ID |

# createLivePlayer

#### Description

This API is used to create a live player instance at the native layer. If you use TXVodPlayerController, as it already integrates an instance, you don't need to create another.





static Future<int?> createLivePlayer() async;

# Parameter description

#### Unlimited

### **Return values**

| Returned Value | Туре | Description   |
|----------------|------|---------------|
| playerId       | int  | The player ID |



# setConsoleEnabled

#### Description

This API is used to enable/disable player native log output.

API



#### static Future<int?> setConsoleEnabled() async;

#### **Parameter description**

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |



| enabled | bool | Enables/Disables player log |
|---------|------|-----------------------------|
|         |      |                             |

# Return values

Unlimited

# releasePlayer

# Description

This API is used to release the player resources.





static Future<int?> releasePlayer(int? playerId) async;

# Parameter description

Unlimited

#### **Return values**

Unlimited

# setGlobalMaxCacheSize

#### Description

This API is used to set the maximum cache size of the player engine. After setting, the backend will clear files in the cache directory automatically according to the set value.





static Future<void> setGlobalMaxCacheSize(int size) async;

# Parameter description

| Parameter | Туре | Description                  |
|-----------|------|------------------------------|
| size      | int  | The maximum cache size in MB |

#### **Return values**

Unlimited

# setGlobalCacheFolderPath

#### Description

This API is used to set the cache path, which is set to the application's sandbox directory by default. You only need to pass in the relative cache directory instead of the entire absolute path to the parameter.

API



static Future<bool> setGlobalCacheFolderPath(String postfixPath) async;

#### Parameter description

Parameter

Description

Type



|             |        | The cache path, which is set to the application's sandbox directory by default. You only need to pass in the relative cache directory instead of the entire absolute path to |                   |           |                    |                        |
|-------------|--------|--|-------------------|-----------|--------------------|------------------------|
| postfixPath | String | postfixPat   | . On Android, vie | deos will | be cached to the   | Android/data/your-     |
|             |        | pkg-name/fi  | lles/testCache    | director  | ry on the SD card. | On iOS, videos will be |
|             |        | cached to the  | Documents/test    | Cache     | directory in the s | andbox.                |
|             |        |  |                   |           |                    |                        |

#### **Return values**

Unlimited

# setLogLevel

# Description

This API is used to set the log output level.





static Future<void> setLogLevel(int logLevel) async;

# Parameter description

| Parameter | Туре | Description  |  |
|-----------|------|--|--|
| logLevel  | int  | <ol> <li>Logs at all levels;</li> <li>: DEBUG, INFO, WARNING, ERROR, and FATAL logs;</li> <li>: INFO, WARNING, ERROR, and FATAL logs;</li> <li>: WARNING, ERROR, and FATAL logs;</li> <li>: ERROR and FATAL logs;</li> </ol> |  |



| 5 : Only FATAL logs; |
|----------------------|
| 6 : No SDK logs.     |

#### **Return values**

Unlimited

# setBrightness

#### Description

This API is used to set the brightness. It is applicable only to the current application.





static Future<void> setBrightness(double brightness) async;

# Parameter description

| Parameter  | Туре   | Description                                |
|------------|--------|--|
| brightness | double | The brightness level. Value range: 0.0-1.0 |

#### **Return values**

Unlimited

# restorePageBrightness

### Description

This API is used to reset the UI brightness. It is applicable only to the current application.

API



static Future<void> restorePageBrightness() async;

Parameter description Unlimited Return values



Unlimited

# getBrightness

# Description

This API is used to get the brightness level of the current UI.

API



static Future<double> getBrightness() async;

Parameter description



#### Unlimited

#### **Return values**

| Parameter  | Туре   | Description                                |
|------------|--------|--|
| brightness | double | The brightness level. Value range: 0.0-1.0 |

# setSystemVolume

# Description

This API is used to set the volume level of the current system.





static Future<void> setSystemVolume(double volume) async;

# Parameter description

| Parameter | Туре   | Description                            |
|-----------|--------|--|
| volume    | double | The volume level. Value range: 0.0-1.0 |

#### **Return values**

Unlimited



# getSystemVolume

### Description

This API is used to set the volume level of the current system.

API



static Future<double> getSystemVolume() async;

Parameter description
Unlimited

Returned value description



| Parameter | Туре   | Description                            |
|-----------|--------|--|
| volume    | double | The volume level. Value range: 0.0-1.0 |

# abandonAudioFocus

# Description

This API is used to release the audio focus. It is applicable only to Android.

#### API



static Future<double> abandonAudioFocus() async;



# **Parameter description**

Unlimited

# **Return values**

Unlimited

# requestAudioFocus

#### Description

This API is used to request the audio focus. It is applicable only to Android.





static Future<void> requestAudioFocus() async ;

#### Parameter description

Unlimited

#### **Return values**

Unlimited

# isDeviceSupportPip

# Description

This API is used to check whether the current device supports the picture-in-picture (PiP) mode.





static Future<int> isDeviceSupportPip() async;

# Parameter description

Unlimited

# **Returned value description**

| Parameter          | Туре | Description   |
|--------------------|------|---|
| isDeviceSupportPip | int  | <ul><li>0 : The PiP mode can be enabled;</li><li>-101 : The Android version is too early;</li></ul> |



| -102 | : The PiP permission is disabled or the device doesn't support PiP; |
|------|---|
| -103 | : The current UI has been terminated.                               |

# getLiteAVSDKVersion

# Description

This API is used to get the version number of the current native-layer player SDK.

API



static Future<String?> getLiteAVSDKVersion() async;

#### Parameter description

#### Unlimited

#### **Returned value description**

| Parameter  | Туре   | Description                            |
|------------|--------|--|
| sdkVersion | String | The version of the current player SDK. |

# startVideoOrientationService

#### Description

Start monitoring the device's rotation direction. Once enabled, if the device's auto-rotate feature is turned on, the player will automatically rotate the video direction based on the current device orientation.

This interface currently only applies to the Android platform. The iOS platform will automatically enable this feature.

# Note

Before calling this interface, please be sure to inform the user of the privacy risks.





static Future<bool> startVideoOrientationService() async

# Parameter description

Unlimited

# **Returned value description**

| Parameter | Туре | Description  |
|-----------|------|--|
| result    | bool | true means the feature was successfully enabled, while false means it failed to enable.<br>Possible reasons for failure include premature activation before the context is<br>initialized or failure to obtain the sensor. |

# registerSysBrightness

#### Description

Enable or disable the monitoring of system brightness. If enabled, when the system brightness changes, it will change the current window brightness and callback the brightness to the Flutter layer. This interface needs to be used in conjunction with setBrightness and onExtraEventBroadcast.

API



static Future<void>registerSysBrightness(bool isRegister) async



# Parameter description

| Parameter  | Туре | Description  |
|------------|------|--|
| isRegister | bool | true: Turn on listening.<br>false: Turns off monitoring. |

# **Returned value description**

Unlimited

# $\texttt{TXVodPlayerController} \ \textbf{Class}$

### initialize

#### Description

This API is used to initialize the controller and request assignment of shared textures.



# Future<void> initialize({bool? onlyAudio}) async;

# Parameter description

| Parameter | Туре | Description   |
|-----------|------|---|
| onlyAudio | bool | Whether the player is an audio-only player. This parameter is optional. |

# Returned value description

Unlimited



# startVodPlay

#### Note

Starting from v10.7, startPlay is replaced by startVodPlay, and playback will succeed only after you use {@link SuperPlayerPlugin#setGlobalLicense} to set the license; otherwise, playback will fail (black screen occurs). The license needs to be set only once globally. You can use the license for CSS, UGSV, or video playback. If you have no such licenses, you can quickly apply for a trial license or purchase an official license. **Description** 

This API is used to play back a video via URL.



Future<bool> startVodPlay(String url) async;

#### Parameter description

| Parameter | Туре   | Description                             |
|-----------|--------|---|
| url       | String | The URL of the video to be played back. |

#### **Returned value description**

| Parameter | Туре | Description                 |
|-----------|------|-----------------------------|
| result    | bool | Whether creation succeeded. |

#### startVodPlayWithParams

#### Note

Starting from v10.7, startPlay is replaced by startVodPlay, and playback will succeed only after you use {@link SuperPlayerPlugin#setGlobalLicense} to set the license; otherwise, playback will fail (black screen occurs). The license needs to be set only once globally. You can use the license for CSS, UGSV, or video playback. If you have no such licenses, you can quickly apply for a trial license or purchase an official license.

#### Description

This API is used to play back a video via fileId .





# Future<void> startVodPlayWithParams(TXPlayInfoParams params) async;

# Parameter description

| Parameter | Туре   | Description   |  |
|-----------|--------|---|--|
| appld     | int    | The application's appld, which is required.                           |  |
| fileId    | String | The file ID, which is required.                                       |  |
| sign      | String | The hotlink protection signature. For more information, see Overview. |  |
|           |        |   |  |



url

String

Video url, either this field or fileld can be filled.

# **Returned value description**

Unlimited

pause

# Description

This API is used to pause a video during playback.




Future<void> pause() async;

# Parameter description

Unlimited

## **Returned value description**

Unlimited

#### resume

#### Description

This API is used to resume the playback of a paused video.



Future<void> resume() async;

## Parameter description

Unlimited

# Returned value description

Unlimited

## stop

Description



This API is used to stop a video during played back.

## API



Future<bool> stop({bool isNeedClear = false}) async;

# Parameter description

| Parameter   | Туре | Description                            |
|-------------|------|--|
| isNeedClear | bool | Whether to clear the last-frame image. |



## Returned value description

| Parameter | Туре | Description             |
|-----------|------|-------------------------|
| result    | bool | Whether stop succeeded. |

# setIsAutoPlay

## Description

This API is used to set whether to automatically play back the video after calling startVodPlay to load the video URL.

API

©2013-2022 Tencent Cloud. All rights reserved.





## Future<void> setIsAutoPlay({bool? isAutoPlay}) async;

## Parameter description

| Parameter  | Туре | Description                                   |
|------------|------|---|
| isAutoPlay | bool | Whether to play back the video automatically. |

## Returned value description

Unlimited



# isPlaying

## Description

This API is used to query whether the player is currently playing a video.

API



Future<bool> isPlaying() async;

Parameter description Unlimited



| Parameter | Туре | Description                  |
|-----------|------|------------------------------|
| isPlaying | bool | Whether playback is ongoing. |

## setMute

# Description

This API is used to set whether to mute the current playback.

#### API



Future<void> setMute(bool mute) async;



## Parameter description

| Parameter | Туре | Description                   |
|-----------|------|-------------------------------|
| mute      | bool | Whether to mute the playback. |

## Returned value description

Unlimited

#### setLoop

## Description

This API is used to specify whether to loop the video after the video playback ends.



Future<void> setLoop(bool loop) async;

# Parameter description

| Parameter | Туре | Description                |
|-----------|------|----------------------------|
| Іоор      | bool | Whether to loop the video. |

## Returned value description

Unlimited



#### seek

## Description

This API is used to adjust the playback progress to the specified time.

API



#### \_controller.seek(progress);

## Parameter description

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |





progress

double

# **Returned value description**

Unlimited

# setRate

# Description

This API is used to set the playback rate.



Future<void> setRate(double rate) async;

#### Parameter description

| •         |        |   |
|-----------|--------|---|
| Parameter | Туре   | Description                             |
| rate      | double | Video playback rate. Default value: 1.0 |

#### **Returned value description**

Unlimited

## getSupportedBitrates

#### Description

This API is used to get the bitrates supported by the video being played back.





Future<List?> getSupportedBitrates() async;

# Parameter description

Unlimited

| Returned Value | Туре | Description                     |
|----------------|------|---------------------------------|
| index          | int  | Bitrate number                  |
| width          | int  | The video width for the bitrate |



| height  | int | Video height for the bitrate |
|---------|-----|------------------------------|
| bitrate | int | Bitrate value                |

# getBitrateIndex

## Description

This API is used to get the set bitrate number.





Future<int> getBitrateIndex() async;

#### Parameter description

Unlimited

## Returned value description

| Returned Value | Туре | Description    |
|----------------|------|----------------|
| index          | int  | Bitrate number |

## setBitrateIndex

#### Description

This API is used to set the current bitrate by bitrate number.





Future<void> setBitrateIndex(int index) async;

# Parameter description

| Returned Value | Туре | Description   |
|----------------|------|---|
| index          | int  | The bitrate number1 indicates to enable adaptive bitrate streaming. |



Unlimited

## setStartTime

#### Description

This API is used to specify the playback start time.

API



Future<void> setStartTime(double startTime) async;

## Parameter description



| Returned Value | Туре   | Description                         |
|----------------|--------|-------------------------------------|
| startTime      | double | The playback start time in seconds. |

# **Returned value description**

Unlimited

# setAudioPlayoutVolume

## Description

This API is used to set the video volume level.



## Future<void> setAudioPlayoutVolume(int volume) async;

## Parameter description

| Parameter | Туре | Description                            |
|-----------|------|--|
| volume    | int  | Video volume level. Value range: 0–100 |

## Returned value description

Unlimited

# setRequestAudioFocus

#### Description

This API is used to set the audio focus. It is applicable only to Android.

API



Future<bool> setRequestAudioFocus(bool focus) async;

#### **Parameter description**

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |



| focus | bool | Whether to set the audio focus. |
|-------|------|---------------------------------|
|       |      |                                 |

# **Returned value description**

| Parameter | Туре | Description                            |
|-----------|------|--|
| result    | bool | Whether the focus is set successfully. |

# setConfig

## Description

This API is used to configure the player.





## Future<void> setConfig(FTXVodPlayConfig config) async ;

# Parameter description

| Parameter | Туре             | Description   |
|-----------|------------------|---|
| config    | FTXVodPlayConfig | For more information, see the FTXVodPlayConfig class. |

## Returned value description

Unlimited

# getCurrentPlaybackTime

#### Description

This API is used to get the current playback time in seconds.

API



Future<double> getCurrentPlaybackTime() async;

Parameter description
Unlimited

# 🔗 Tencent Cloud

| Parameter    | Туре   | Description                           |
|--------------|--------|---------------------------------------|
| playbackTime | double | The current playback time in seconds. |

# getBufferDuration

# Description

This API is used to get the currently buffered video duration in seconds.

#### API



Future<double> getBufferDuration();

## Parameter description

#### Unlimited

# **Returned value description**

| Parameter    | Туре   | Description                                       |
|--------------|--------|---|
| playbackTime | double | The currently buffered video duration in seconds. |

## getPlayableDuration

#### Description

This API is used to get the playable duration of the video being played back in seconds.





Future<double> getPlayableDuration() async;

# Parameter description

Unlimited

| Parameter        | Туре   | Description                                       |
|------------------|--------|---|
| playableDuration | double | The currently playable video duration in seconds. |



## getWidth

#### Description

This API is used to get the width of the video being played back.

API



Future<int> getWidth() async;

Parameter description

Unlimited



| Parameter | Туре | Description             |
|-----------|------|-------------------------|
| width     | int  | The current video width |

# getHeight

# Description

This API is used to get the height of the video being played back.

#### API



Future<int> getHeight() async;

## Parameter description

Unlimited

# Returned value description

| Parameter | Туре | Description               |
|-----------|------|---------------------------|
| height    | int  | The current video height. |

## setToken

#### Description

This API is used to set the token for HLS encryption. After the token is set, the player will automatically add

voddrm.token before the filename in the URL.



Future<void> setToken(String? token) async;

# Parameter description

| Parameter | Туре   | Description                   |
|-----------|--------|-------------------------------|
| token     | String | The token for video playback. |

## Returned value description

Unlimited



# isLoop

#### Description

This API is used to get the current playback loop status of the player.

API



Future<bool> isLoop() async;

Parameter description Unlimited



| Parameter | Туре | Description                           |
|-----------|------|---------------------------------------|
| isLoop    | bool | Whether the player is in loop status. |

## enableHardwareDecode

## Description

This API is used to enable/disable playback based on hardware decoding. After the value is set, it will not take effect until the video playback is restarted.



Future<bool> enableHardwareDecode(bool enable);

#### Parameter description

| Parameter | Туре | Description                          |
|-----------|------|--------------------------------------|
| enable    | bool | Whether to enable hardware decoding. |

#### **Returned value description**

| Parameter | Туре | Description                                    |
|-----------|------|--|
| result    | bool | The hardware/software decoding setting result. |

## dispose

#### Description

This API is used to terminate the controller. After it is called, all notification events will be terminated, and the player will be released.



Future<void> dispose() async;

## Parameter description

Unlimited

# Returned value description

Unlimited

# getDuration

Description



This API is used to get the total video duration.

## API



Future<double> getDuration() async;

# Parameter description

Unlimited

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |





|  | duration | double | The total video duration in seconds. |
|--|----------|--------|--------------------------------------|
|--|----------|--------|--------------------------------------|

# enterPictureInPictureMode

# Description

This API is used to enter the PiP mode.

API



Future<int> enterPictureInPictureMode({String? backIconForAndroid, String? playIcon

## Parameter description
The parameters are applicable only to Android.

| Parameter   | Туре   | Description  |
|-------------|--------|--|
| backlcon    | String | The seek backward icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used. |
| playlcon    | String | The playback icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used.      |
| pauselcon   | String | The pause icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used.         |
| forwardlcon | String | The fast forward icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used.  |

### Returned value description

| Parameter                        | Code | Description   |
|----------------------------------|------|---|
| NO_ERROR                         | 0    | Started successfully with no errors.  |
| ERROR_PIP_LOWER_VERSION          | -101 | The Android version is too early and doesn't support the PiP mode.  |
| ERROR_PIP_DENIED_PERMISSION      | -102 | The PiP mode permission wasn't enabled, or the current device doesn't support PiP.  |
| ERROR_PIP_ACTIVITY_DESTROYED     | -103 | The current UI was terminated.  |
| ERROR_IOS_PIP_DEVICE_NOT_SUPPORT | -104 | The device model or system version doesn't support PiP (only supported on iPadOS 9+ and iOS 14+). This error is applicable only to iOS. |
| ERROR_IOS_PIP_PLAYER_NOT_SUPPORT | -105 | The player doesn't support PiP. This error is applicable only to iOS.   |
| ERROR_IOS_PIP_VIDEO_NOT_SUPPORT  | -106 | The video doesn't support PiP. This error is applicable only to iOS.  |
| ERROR_IOS_PIP_IS_NOT_POSSIBLE    | -107 | The PiP controller was unavailable. This error is applicable only to iOS.   |
| ERROR_IOS_PIP_FROM_SYSTEM        | -108 | The PiP controller reported an error. This error is applicable only to iOS.   |
| ERROR_IOS_PIP_PLAYER_NOT_EXIST   | -109 | The player object doesn't exist. This error is applicable only to iOS.  |

| ERROR_IOS_PIP_IS_RUNNING  | -110 | The PiP feature was running. This error is applicable only to iOS.  |
|---------------------------|------|---|
| ERROR_IOS_PIP_NOT_RUNNING | -111 | The PiP feature didn't start. This error is applicable only to iOS. |

# initImageSprite

# Description

Initialize video sprite image





Future<void> initImageSprite(String? vvtUrl, List<String>? imageUrls) async;

# Parameter description

| Parameter | Туре                   | Description   |
|-----------|------------------------|---|
| vvtUrl    | String                 | Sprite Image web VTT description file download URL. |
| imageUrls | List <string></string> | Sprite image download URL.                          |



### Returned value description

Unlimited

# getImageSprite

# Description

Get the loaded sprite Image.

API



Future<Uint8List?> getImageSprite(double time) async;



### Parameter description

| Parameter | Туре   | Description             |
|-----------|--------|-------------------------|
| time      | double | Time point, in seconds. |

# Returned value description

| Parameter | Туре      | Description  |
|-----------|-----------|--------------|
| thumb     | Uint8List | Sprite Image |

### exitPictureInPictureMode

### Description

Exit picture-in-picture mode if the player is currently in picture-in-picture mode.



Future<void> exitPictureInPictureMode() async;

### Parameter description

Unlimited

# Returned value description

Unlimited

### addSubtitleSource

Description



Add external subtitles.

Note: This feature requires Player Premium version 11.7 to be supported.

API



Future<void> addSubtitleSource(String url, String name, {String? mimeType}) async;

### Parameter description

| Parameter | Туре   | Description  |
|-----------|--------|--------------|
| url       | String | subtitle url |
|           |        |              |



| name     | String | subtitle name   |
|----------|--------|---|
| mimeType | String | Subtitle type, supports SRT<br>(TXVodPlayEvent.VOD_PLAY_MIMETYPE_TEXT_SRT) and VVT<br>(TXVodPlayEvent.VOD_PLAY_MIMETYPE_TEXT_VTT) formats |

# getSubtitleTrackInfo

### Description

Returns the subtitle track information list.

Note: This feature requires Player Premium version 11.7 to be supported.





### Future<List<TXTrackInfo>> getSubtitleTrackInfo() async;

# Parameter description

TXTrackInfo class :

| Parameter | Туре | Description   |
|-----------|------|---|
| trackType | int  | Track type. The values are:<br>Video track: TX_VOD_MEDIA_TRACK_TYPE_VIDEO = 1<br>Audio track: TX_VOD_MEDIA_TRACK_TYPE_AUDIO = 2<br>Subtitle track: TX_VOD_MEDIA_TRACK_TYPE_SUBTITLE = 3 |

| trackIndex  | int    | track index   |
|-------------|--------|---|
| name        | String | track name  |
| isSelected  | bool   | Whether the current track is selected   |
| isExclusive | bool   | If it is true, only one track of this type can be selected at each time. If it is false, multiple tracks of this type can be selected at the same time. |
| isInternal  | bool   | Whether the current track is an internal original track.  |

# getAudioTrackInfo

### Description

Returns the subtitle track information list.

Note: This feature requires Player Premium version 11.7 to be supported.





Future<List<TXTrackInfo>> getAudioTrackInfo() async;

### Parameter description

Reference TXTrackInfo class

### selectTrack

#### Description

Select track.

Note: This feature requires Player Premium version 11.7 to be supported.



API



### Future<void> selectTrack(int trackIndex) async;

# Parameter description

| Parameter  | Туре | Description  |
|------------|------|--|
| trackIndex | int  | Track index, trackIndex track index, obtained through the trackIndex of [TXTrackInfo]. |



### deselectTrack

### Description

Deselect the track.

Note: This feature requires Player Premium version 11.7 to be supported.

API



Future<void> deselectTrack(int trackIndex) async;

| Parameter description |      |             |  |
|-----------------------|------|-------------|--|
| Parameter             | Туре | Description |  |



| trackIndex int Track index, trackIndex track index, obtained through the track [TXTrackInfo]. | Index of |
|---|----------|
|---|----------|

# FTXVodPlayConfig Class

### Attribute configuration description

| Parameter            | Туре   | Description  |
|----------------------|--------|--|
| connectRetryCount    | int    | The number of player reconnections. If the SDK is disconnected from the server due to an exception, the SDK will attempt to reconnect to the server.   |
| connectRetryInterval | int    | The interval between two player reconnections. If the SDK is disconnected from the server due to an exception, the SDK will attempt to reconnect to the server.  |
| timeout              | int    | Player connection timeout period   |
| playerType           | int    | Player type. Valid values: 0: VOD; 1: live streaming; 2: live stream replay.   |
| headers              | Мар    | Custom HTTP headers  |
| enableAccurateSeek   | bool   | Whether to enable accurate seek. Default value: true .   |
| autoRotate           | bool   | If it is set to true, the MP4 file will be automatically rotated according to the rotation angle set in the file, which can be obtained from the PLAY_EVT_CHANGE_ROTATION event. Default value: true.                    |
| smoothSwitchBitrate  | bool   | Whether to enable smooth multi-bitrate HLS stream switch. If it is set to false (default), multi-bitrate URLs are opened faster. If it is set to true, the bitrate can be switched smoothly when IDR frames are aligned. |
| cacheMp4ExtName      | String | The cached MP4 filename extension. Default value: mp4 .  |
| progressInterval     | int    | The progress callback interval in milliseconds. If it is not set, the SDK will call back the progress once every 0.5 seconds.  |
| maxBufferSize        | int    | The maximum size of playback buffer in MB. The setting will affect playableDuration. The greater the value, the more the data that is buffered in advance.   |
| maxPreloadSize       | int    | Maximum preload buffer size in MB  |
|                      |        |  |



| firstStartPlayBufferTime | int    | Duration of the video data that needs to be loaded during the first buffering in ms. Default value: 100 ms   |
|--------------------------|--------|--|
| nextStartPlayBufferTime  | int    | Minimum buffered data size to stop buffering (secondary buffering for insufficient buffered data or progress bar drag buffering caused by seek) in ms. Default value: 250 ms   |
| overlayKey               | String | The HLS security enhancement encryption and decryption key   |
| overlaylv                | String | The HLS security enhancement encryption and decryption IV  |
| extInfoMap               | Мар    | Some special configuration items   |
| enableRenderProcess      | bool   | Whether to allow the postrendering and postproduction feature, which is<br>enabled by default. If the super-resolution plugin exists after it is enabled,<br>the plugin will be loaded by default  |
| preferredResolution      | int    | Resolution of the video used for playback preferably.<br>preferredResolution = width * height  |
| mediaType                | int    | Set the media type, with AUTO as the default value. Optional values are:<br>TXVodConstants#MEDIA_TYPE_AUTO, AUTO type (default value,<br>adaptive bit rate playback is not supported for now);<br>TXVodConstants#MEDIA_TYPE_HLS_VOD, HLS VOD media;<br>TXVodConstants#MEDIA_TYPE_HLS_LIVE, HLS live media;<br>TXVodConstants#MEDIA_TYPE_HLS_VOD, MP4 and other common<br>file VOD media (supported starting from version 11.7);<br>TXVodConstants#MEDIA_TYPE_DASH_VOD, DASH VOD media<br>(supported starting from version 11.7); |

# TXLivePlayerController Class

### initialize

### Description

This API is used to initialize the controller and request assignment of shared textures.

🕗 Tencent Cloud



# Future<void> initialize({bool? onlyAudio}) async;

# Parameter description

| Parameter | Туре | Description   |
|-----------|------|---|
| onlyAudio | bool | Whether the player is an audio-only player. This parameter is optional. |

### **Returned value description**

Unlimited



# startLivePlay

#### Note

Starting from v10.7, startPlay is replaced by startLivePlay, and playback will succeed only after you use {@link SuperPlayerPlugin#setGlobalLicense} to set the license; otherwise, playback will fail (black screen occurs). The license needs to be set only once globally. You can use the license for CSS, UGSV, or video playback. If you have no such licenses, you can quickly apply for a trial license or purchase an official license. **Description** 

This API is used to play back a video via URL.



### Future<bool> play(String url, {int? playType}) async;

### **Parameter description**

| Parameter | Туре   | Description   |
|-----------|--------|---|
| url       | String | The URL of the video to be played back.   |
| playType  | int    | The supported playback type, which is optional. RTMP live streaming is used by default. Valid values: LIVE_RTMP, LIVE_FLV, LIVE_RTMP_ACC, and |

### **Returned value description**

| Parameter | Туре | Description                 |
|-----------|------|-----------------------------|
| result    | bool | Whether creation succeeded. |

### pause

### Description

This API is used to pause a video during playback.



Future<void> pause() async;

# Parameter description

Unlimited

# Returned value description

Unlimited

#### resume

Description



This API is used to resume the playback of a paused video.

### API



Future<void> resume() async;

Parameter description Unlimited Returned value description Unlimited



### stop

### Description

This API is used to stop a video during played back.

API



#### Future<bool> stop({bool isNeedClear = false}) async;

### Parameter description

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |



| isNeedClear | bool | Whether to clear the last-frame image. |
|-------------|------|--|
|             |      |  |

# **Returned value description**

| Parameter | Туре | Description             |
|-----------|------|-------------------------|
| result    | bool | Whether stop succeeded. |

# isPlaying

### Description

This API is used to query whether the player is currently playing a video.



Future<bool> isPlaying() async;

# Parameter description

Unlimited

# **Returned value description**

| Parameter | Туре | Description                  |
|-----------|------|------------------------------|
| isPlaying | bool | Whether playback is ongoing. |



### setMute

### Description

This API is used to set whether to mute the current playback.

API



### Future<void> setMute(bool mute) async;

### Parameter description

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |



mute

bool

# Returned value description

Unlimited

# setVolume

# Description

This API is used to set the video volume level.



Future<void> setVolume(int volume);

#### Parameter description

| Parameter | Туре | Description                            |
|-----------|------|--|
| volume    | int  | Video volume level. Value range: 0-100 |

### **Returned value description**

Unlimited

# setLiveMode

### Description

This API is used to set the live streaming mode.





Future<void> setLiveMode(TXPlayerLiveMode mode) async;

### Parameter description

| Parameter | Туре | Description   |
|-----------|------|---|
| mode      | int  | Live streaming mode, which can be set to auto, expedited, or smooth mode. |

### Returned value description

Unlimited



# setAppID

### Description

This API is used to set the appID for cloud-based control.

API



#### Future<void> setAppID(int appId) async;

### Parameter description

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |



| appld ir | int | The application ID. |
|----------|-----|---------------------|
|          |     |                     |

# **Returned value description**

Unlimited

# setConfig

# Description

This API is used to configure the player.





Future<void> setConfig(FTXLivePlayConfig config) async;

#### Parameter description

| Parameter | Туре              | Description                   |                   |        |
|-----------|-------------------|-------------------------------|-------------------|--------|
| config    | FTXLivePlayConfig | For more information, see the | FTXLivePlayConfig | class. |

### **Returned value description**

Unlimited

### enableHardwareDecode

### Description

This API is used to enable/disable playback based on hardware decoding. After the value is set, it will not take effect until the video playback is restarted.



Future<bool> enableHardwareDecode(bool enable);

# Parameter description

| Parameter | Туре | Description                          |
|-----------|------|--------------------------------------|
| enable    | bool | Whether to enable hardware decoding. |

### **Returned value description**

| Parameter | Туре | Description |
|-----------|------|-------------|
|-----------|------|-------------|



result

bool

The hardware/software decoding setting result.

### enterPictureInPictureMode

### Description

This API is used to enter the PiP mode. It is applicable only to Android. Currently, live streaming on iOS doesn't support the PiP mode.

### API



Future<int> enterPictureInPictureMode({String? backIconForAndroid, String? playIcon

# Parameter description

The parameters are applicable only to Android.

| Parameter   | Туре   | Description  |
|-------------|--------|--|
| backlcon    | String | The seek backward icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used. |
| playIcon    | String | The playback icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used.      |
| pauselcon   | String | The pause icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used.         |
| forwardlcon | String | The fast forward icon, which can be up to 1 MB in size as limited by Android. It is optional, and if it is not set, the system icon will be used.  |

# Returned value description

| Parameter                        | Code | Description   |
|----------------------------------|------|---|
| NO_ERROR                         | 0    | Started successfully with no errors.  |
| ERROR_PIP_LOWER_VERSION          | -101 | The Android version is too early and doesn't support the PiP mode.  |
| ERROR_PIP_DENIED_PERMISSION      | -102 | The PiP mode permission wasn't enabled, or the current device doesn't support PiP.  |
| ERROR_PIP_ACTIVITY_DESTROYED     | -103 | The current UI was terminated.  |
| ERROR_IOS_PIP_DEVICE_NOT_SUPPORT | -104 | The device model or system version doesn't support PiP (only supported on iPadOS 9+ and iOS 14+). This error is applicable only to iOS. |
| ERROR_IOS_PIP_PLAYER_NOT_SUPPORT | -105 | The player doesn't support PiP. This error is applicable only to iOS.   |
| ERROR_IOS_PIP_VIDEO_NOT_SUPPORT  | -106 | The video doesn't support PiP. This error is applicable only to iOS.  |
| ERROR_IOS_PIP_IS_NOT_POSSIBLE    | -107 | The PiP controller was unavailable. This error is applicable only to iOS.   |
| ERROR_IOS_PIP_FROM_SYSTEM        | -108 | The PiP controller reported an error. This error is applicable only to iOS.   |
| ERROR_IOS_PIP_PLAYER_NOT_EXIST   | -109 | The player object doesn't exist. This error is  |



|                           |      | applicable only to iOS.   |
|---------------------------|------|---|
| ERROR_IOS_PIP_IS_RUNNING  | -110 | The PiP feature was running. This error is applicable only to iOS.  |
| ERROR_IOS_PIP_NOT_RUNNING | -111 | The PiP feature didn't start. This error is applicable only to iOS. |

# dispose

### Description

This API is used to terminate the controller. After it is called, all notification events will be terminated, and the player will be released.



Future<void> dispose() async;

# Parameter description

Unlimited

# Returned value description

Unlimited

### switchStream

Description



This API is used to switch the stream played back.

### API



Future<int> switchStream(String url) async;

### Parameter description

| Parameter | Туре   | Description                         |
|-----------|--------|-------------------------------------|
| url       | String | The video source to be switched to. |


# Returned value description

| Parameter | Туре | Description        |
|-----------|------|--------------------|
| result    | int  | The switch result. |

# FTXLivePlayConfig Class

# Attribute configuration description

| Parameter              | Туре   | Description   |
|------------------------|--------|---|
| cacheTime              | double | The player cache time in seconds. The value must be greater than          0       . Default value:       5       .  |
| maxAutoAdjustCacheTime | double | The maximum time for automatic cache time adjustment in seconds.<br>The value must be greater than 0 . Default value: 5 .   |
| minAutoAdjustCacheTime | double | The minimum time for automatic cache time adjustment in seconds.<br>The value must be greater than 0. Default value: 1.   |
| videoBlockThreshold    | int    | The alarm threshold for player video lag in milliseconds. Only when<br>the rendering interval exceeds this threshold, will the<br>PLAY_WARNING_VIDEO_PLAY_LAG notification be sent.   |
| connectRetryCount      | int    | The default number of times the SDK tries reconnecting when the player is disconnected. Value range: 1–10. Default value: 3   |
| connectRetryInterval   | int    | The interval between network reconnections in seconds. Value range: 3–30. Default value: 3  |
| autoAdjustCacheTime    | bool   | Whether to automatically adjust the player cache time. Default value:<br>true . Valid values: true : Enable automatic adjustment. You<br>can use maxCacheTime and minCacheTime to specify the<br>maximum and minimum cache time respectively; false : Disable<br>automatic adjustment and use the default cache time (1s). You can<br>use cacheTime to change the cache time. |
| enableAec              | bool   | Whether to enable acoustic echo cancellation (AEC). Default value: false .  |
| enableMessage          | bool   | Whether to enable the message channel. Default value: true .  |
| enableMetaData         | bool   | Whether to enable theMetaDatacallback. Default value:false. Valid values:true: The SDK throws theMetaData   |



|               |        | of the video stream through theEVT_PLAY_GET_METADATAmessage;false: The SDK doesn't throw theMetaDatavideo stream. |
|---------------|--------|---|
| flvSessionKey | String | Whether to enable HTTP header information callback. Default value:  |

# TXVodDownloadController Class

# startPreLoad

#### Description

Start preloading. Before starting preloading, please set the cache directory

[SuperPlayerPlugin.setGlobalCacheFolderPath] and cache size

[SuperPlayerPlugin.setGlobalMaxCacheSize] of the playback engine. This setting is a global

configuration and needs to be consistent with the player to avoid invalidating the playback cache.



```
Future<int> startPreLoad(
final String playUrl,
final int preloadSizeMB,
final int preferredResolution, {
FTXPredownlodOnCompleteListener? onCompleteListener,
   FTXPredownlodOnErrorListener? onErrorListener,
}) async
```





Future<void> startPreload(TXPlayInfoParams txPlayInfoParams, final int preloadSizeMB, final int preferredResolution, { FTXPredownlodOnCompleteListener? onCompleteListener, FTXPredownlodOnErrorListener? onErrorListener, FTXPredownlodOnStartListener? onStartListener, }) async

#### **Parameter description**

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |



| playUrl             | String                           | The URL to be preloaded  |
|---------------------|----------------------------------|--|
| preloadSizeMB       | int                              | Size of preloaded data (unit: MB).   |
| preferredResolution | int                              | Expected resolution, with the value in the format of height x width. For example, 720x1080. If multiple resolutions are not supported or not specified, pass -1. |
| onCompleteListener  | FTXPredownlodOnCompleteListener? | Preloading successful callback, global.  |
| onErrorListener     | FTXPredownlodOnErrorListener     | Preloading failed callback, global.  |

# TXPlayInfoParams:

| Parameter | Туре   | Description  |
|-----------|--------|--|
| appld     | int    | The application's appld, which is required.  |
| fileId    | String | The file ID, which is required.  |
| url       | String | Only one of the video url and fileId needs to be filled in. If both are filled in, the url takes priority. |
| sign      | String | Hotlink protection signature, see Hotlink protection product documentation.                                |

# **Returned value description**

| Parameter | Туре | Description |
|-----------|------|-------------|
| taskld    | int  | task ID     |

# stopPreLoad

Description

Stop preloading.





Future<void> stopPreLoad(final int taskId) async

# Parameter description

| Parameter | Туре | Description |
|-----------|------|-------------|
| taskld    | int  | task ID     |

# **Returned value description**

Unlimited



# startDownload

# Description

Start downloading the video.

API



Future<void> startDownload(TXVodDownloadMediaInfo mediaInfo) async

# Parameter description

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |



mediaInfo

TXVodDownloadMediaInfo

Download task information.

#### TXVodDownloadMediaInfo

| Parameter        | Туре                     | Description   |
|------------------|--------------------------|---|
| playPath         | String?                  | Cache address. This value is obtained when accessing<br>the video cache and does not need to be assigned<br>when starting the download.   |
| progress         | double?                  | Cache progress. This value is obtained when accessing the video cache and does not need to be assigned when starting the download.  |
| downloadState    | int?                     | Cache status. This value is obtained when accessing<br>the video cache and does not need to be assigned<br>when starting the download.  |
| userName         | String?                  | Download account name used to distinguish<br>downloads from different accounts. Pass an empty<br>string to use "default".   |
| duration         | int?                     | Total duration of cached video. The unit is milliseconds<br>on Android and seconds on iOS. This value is obtained<br>when accessing the video cache and does not need to<br>be assigned when starting the download. |
| playableDuration | int?                     | Cached duration of the video. The unit is milliseconds<br>on Android and seconds on iOS. This value is obtained<br>when accessing the video cache and does not need to<br>be assigned when starting the download.   |
| size             | int?                     | Total file size, in bytes. This value is obtained when accessing the video cache and does not need to be assigned when starting the download.   |
| downloadSize     | int?                     | Downloaded file size, in bytes. This value is obtained<br>when accessing the video cache and does not need to<br>be assigned when starting the download.  |
| url              | String?                  | The URL of the video to be downloaded. This is a required field for downloading URLs. Nesting m3u8 and mp4 downloads is not supported.  |
| dataSource       | TXVodDownloadDataSource? | File ID information of the video to be downloaded.<br>Either the URL or this parameter can be used, but at<br>least one is required.  |



| speed            | int?  | Download speed, in units of KBytes/second.                             |
|------------------|-------|--|
| isResourceBroken | bool? | Whether the resource is corrupted, e.g. the resource has been deleted. |

#### **TXVodDownloadDataSource**

| Parameter | Туре    | Description   |
|-----------|---------|---|
| appId     | int?    | The App ID corresponding to the downloaded file. This is a required field.  |
| fileId    | String? | The ID of the downloaded file. This is a required field.  |
| pSign     | String? | Encryption signature. This is a required field for encrypted videos.  |
| quality   | int?    | The ID of the video quality. This is a required field.  |
| token     | String? | Encryption token.   |
| userName  | String? | Download account name used to distinguish downloads from different accounts. Pass an empty string to use "default". |

# Returned value description

Unlimited

# stopDownload

# Description

Stop downloading.





Future<void> stopDownload(TXVodDownloadMediaInfo mediaInfo) async

# Parameter description

| Parameter | Туре                   | Description       |
|-----------|------------------------|-------------------|
| mediaInfo | TXVodDownloadMediaInfo | Task information. |

# Returned value description

Unlimited

# setDownloadHeaders

### Description

Set the request header for the download task.

API



Future<void> setDownloadHeaders(Map<String, String> headers) async

# **Parameter descriptions**

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |



headers

Map<String, String>

request heade information

# **Returned value description**

Unlimited

# getDownloadList

# Description

Get all download tasks, including completed, in-progress, and failed tasks.



Future<List<TXVodDownloadMediaInfo>> getDownloadList() async

# Parameter descriptions

#### Unlimited

#### **Returned value description**

| Parameter     | Туре   | Description  |
|---------------|--|--|
| mediaInfoList | List <txvoddownloadmediainfo></txvoddownloadmediainfo> | Task list. Different users' downloads can be distinguished by comparing the user name. |

# getDownloadInfo

# Description

Get download task information.





Future<TXVodDownloadMediaInfo> getDownloadInfo(TXVodDownloadMediaInfo mediaInfo) as

# **Parameter descriptions**

| Parameter | Туре                   | Description      |
|-----------|------------------------|------------------|
| mediaInfo | TXVodDownloadMediaInfo | task information |

# **Returned value description**

| Parameter | Туре | Description |
|-----------|------|-------------|
|-----------|------|-------------|



| mediaInfo TXVodDownloadMediaInfo | Cached task details. |
|----------------------------------|----------------------|
|----------------------------------|----------------------|

# setDownloadObserver

# Description

Get download task information.

API



void setDownloadObserver(FTXDownlodOnStateChangeListener downlodOnStateChangeListen

# **Parameter descriptions**

| Parameter                    | Туре                            | Description                    |
|------------------------------|---------------------------------|--------------------------------|
| downlodOnStateChangeListener | FTXDownlodOnStateChangeListener | Task download status callback. |
| downlodOnErrorListener       | FTXDownlodOnErrorListener       | Task download error callback.  |

# **Returned value description**

Unlimited

# deleteDownloadMediaInfo

#### Description

Delete downloaded video.





Future<bool> deleteDownloadMediaInfo(TXVodDownloadMediaInfo mediaInfo) async

# **Parameter descriptions**

| Parameter | Туре                   | Description          |
|-----------|------------------------|----------------------|
| mediaInfo | TXVodDownloadMediaInfo | Cached task details. |

# **Returned value description**

| Parameter | Туре | Description |
|-----------|------|-------------|
|           |      |             |



| result | bool | Deletion result. |
|--------|------|------------------|
|        |      |                  |

Video on Demand

# Player Adapter Player Adapter for iOS

Last updated : 2022-10-17 11:16:01

Player Adapter for iOS is a player plugin provided by VOD for customers who want to use a third-party or proprietary player to connect to Tencent Cloud PaaS resources. It is generally used by customers who strongly need to customize player features.

# SDK Download

The Player Adapter SDK and demo for iOS can be downloaded here.

# Integration Guide

# **Environment requirements**

To configure the support for HTTP requests, you need to set App Transport Security Settings->Allow Arbitrary Loads to YES in the info.plist file of your project.

# **Component dependency**

Add the GCDWebServer component dependency.

pod "GCDWebServer", "~> 3.0"

GCDWebServer is a lightweight HTTP server based on GCD and can be used for OS X and iOS. This library also implements extended features such as web-based file upload and WebDAV server.

# **Using the Player**

Declare the variables and then create an instance. The main class of the player is TXCPlayerAdapter, and videos can be played back after it is created.

- A fileId is usually returned by the server after the video is uploaded:
- 1. After the video is published on the client, the server will return a fileId to the client.
- 2. When the video is uploaded to the server, the corresponding fileId will be included in the notification of upload confirmation.

If the file already exists in Tencent Cloud, you can go to Media Assets and find it. After clicking it, you can view relevant parameters in the video details on the right.

```
NSInteger appId; //// `appid` can be applied for in Tencent Cloud VOD
NSString *fileId;
// `psign` is a player signature. For more information on the signature and how t
o generate it, visit https://cloud.tencent.com/document/product/266/42436.
NSString *pSign = self.pSignTextView.text;
```

TXCPlayerAdapter \*adapter = [TXCPlayerAdapter shareAdapterWithAppId:appId];

Request the video information and play back the video:

```
id<ITXCPlayerAssistorProtocol> assistor = [TXCPlayerAdapter createPlayerAssistorW
ithFileId:fileId pSign:pSign];
[assistor requestVideoInfo:^(id<ITXCPlayerAssistorProtocol> response, NSError *er
ror) {
if (error) {
NSLog(@"create player assistor error : %@",error);
[self.view makeToast:error.description duration: 5.0 position:CSToastPositionBotto
m];
return;
}
[weakSelf avplayerPlay:response]; // Play back the video
}];
- (void) avplayerPlay: (id<ITXCPlayerAssistorProtocol>) response
{
AVPlayerViewController *playerVC = [[AVPlayerViewController alloc] init];
self.playerVC = playerVC;
TXCStreamingInfo *info = response.getStreamingInfo;
AVPlayer *player = [[AVPlayer alloc] initWithURL:[NSURL URLWithString:info.playUr
1]];
playerVC.player = player;
playerVC.title = response.getVideoBasicInfo.name;
[self.navigationController pushViewController:playerVC animated:YES];
[player addObserver:self forKeyPath:@"status" options:NSKeyValueObservingOptionNe
w context:nil];
}
```

Terminate the player after use:

[TXCPlayerAdapter destroy];

# SDK API Description

# **Initializing Adapter**

This API is used to initialize an Adapter singleton.

# API

+ (instancetype) shareAdapterWithAppId: (NSUInteger) appId;

#### Parameter description

appld: Enter the appid (if a subapplication is used, enter the subappid ). |

#### **Terminating Adapter**

This API is used to terminate Adapter. It can be called after the program exits.

#### API

+ (**void**) destroy;

### Creating the Player auxiliary class

An auxiliary class of the player can be used to get the playback fileId and process DRM encryption APIs.

#### API

```
+ (id<ITXCPlayerAssistorProtocol>)createPlayerAssistorWithFileId:(NSString *)file
Id
pSign:(NSString *)pSign;
```

#### **Parameter description**

| Parameter | Туре   | Description                           |
|-----------|--------|---------------------------------------|
| fileId    | String | The fileId of the video to be played. |
| pSign     | String | The player signature.                 |

#### **Requesting video playback information**

This API is used to request the stream information of the video to be played back from the Tencent Cloud VOD server.



(void) requestVideoInfo: (ITXCRequestVideoInfoCallback) completion;

#### Parameter description

| Parameter  | Туре                         | Description              |
|------------|------------------------------|--------------------------|
| completion | ITXCRequestVideoInfoCallback | Async callback function. |

#### Terminating the Player auxiliary class

This API is used to terminate an auxiliary class. You can call it when exiting the player or switching to the next video for playback.

#### API

+ (void) destroyPlayerAssistor: (id<ITXCPlayerAssistorProtocol>) assistor;

#### Getting the basic video information

This API is used to get the video information and will take effect only after

```
id<itxcplayerassistorprotocol>.requestVideoInfo is called back.
```

#### API

- (TXCVideoBasicInfo \*)getVideoBasicInfo;

#### Parameter description

The parameters of TXCVideoBasicInfo are as follows:

| Parameter   | Туре   | Description               |
|-------------|--------|---------------------------|
| name        | String | Video name                |
| size        | Int    | Video size in bytes       |
| duration    | Float  | Video duration in seconds |
| description | String | Video description         |
| coverUrl    | String | Video cover               |

#### Getting video stream information

#### This API is used to get the video stream information list and will take effect only after

 $\verb"id<" txcplayerassistor<" torcol>.requestVideoInfo" is called back.$ 

#### API

- (TXCStreamingInfo \*)getStreamingInfo;

#### **Parameter description**

The parameters of TXCStreamingInfo are as follows:

| Parameter  | Туре   | Description   |
|------------|--------|---|
| playUrl    | String | Playback URL  |
| subStreams | List   | The adaptive bitrate substream information of the SubStreamInfo type. |

The parameters of TXCSubStreamInfo are as follows:

| Parameter      | Туре   | Description  |
|----------------|--------|--|
| type           | String | Substream type. Valid values: video                      |
| width          | Int    | The substream video width in px.                         |
| height         | Int    | The substream video height in px.                        |
| resolutionName | String | The name of the substream video displayed in the player. |

# Getting keyframe timestamp information

This API is used to get the video keyframe timestamp information and will take effect only after

id<itxcplayerassistorprotocol>.requestVideoInfo is called back.

# API

- (NSArray<TXCKeyFrameDescInfo \*> \*)getKeyFrameDescInfos;

# Parameter description

The parameters of TXCKeyFrameDescInfo are as follows:

| Parameter  | Туре  | Description |
|------------|-------|-------------|
| timeOffset | Float | 1.1         |



| Parameter | Туре   | Description     |
|-----------|--------|-----------------|
| content   | String | "Beginning now" |

# Getting thumbnail information

This API is used to get the thumbnail information and will take effect only after

```
id<itxcplayerassistorprotocol>.requestVideoInfo is called back.
```

#### API

```
- (TXCImageSpriteInfo *)getImageSpriteInfo;
```

#### **Parameter description**

The parameters of TCXImageSpriteInfo are as follows:

| Parameter | Туре   | Description                                      |
|-----------|--------|--|
| imageUrls | List   | Array of thumbnail download URLs of String type. |
| webVttUrl | String | Thumbnail VTT file download URL.                 |
|           |        |  |

# Player Adapter for Android

Last updated : 2022-10-17 11:16:01

Player Adapter for Android is a player plugin provided by VOD for customers who want to use a third-party or proprietary player to connect to Tencent Cloud PaaS resources. It is generally used by customers who strongly need to customize player features.

# SDK Download

The Player Adapter SDK and demo for Android can be downloaded here.

# Integration Guide

# Integrating the SDK

Integrate the SDK, copy TXCPlayerAdapter-release-1.0.0.aar to the libs directory, and add dependencies:

implementation(name:'TXCPlayerAdapter-release-1.0.0', ext:'aar')

Add the script for obfuscation:

```
-keep class com.tencent.** { *;}
```

# **Using the Player**

Declare the variables and then create an instance. The main class of the player is ITXCPlayerAssistor, and videos can be played back after it is created.

- A fileId is usually returned by the server after the video is uploaded:
- 1. After the video is published on the client, the server will return a fileId to the client.
- 2. When the video is uploaded to the server, the corresponding fileId will be included in the notification of upload confirmation.

If the file already exists in Tencent Cloud, you can go to Media Assets and find it. After clicking it, you can view relevant parameters in the video details on the right.

🔗 Tencent Cloud

// `psign` is a player signature. For more information on the signature and how t
o generate it, visit https://cloud.tencent.com/document/product/266/42436.
private String mFileId, mPSign;
ITXCPlayerAssistor mPlayerAssistor = TXCPlayerAdapter.createPlayerAssistor(mFileI
d, mPSign);

Initialization:

```
// Initialize the component
TXCPlayerAdapter.init(appId); // `appid` can be applied for in Tencent Cloud VOD
TXCPlayerAdapter.setLogEnable(true); // Enable log
mSuperPlayerView = findViewById(R.id.sv_videoplayer);
mPlayerAssistor = TXCPlayerAdapter.createPlayerAssistor(mFileId, mPSign);
```

Request the video information and play back the video:

```
mPlayerAssistor.requestVideoInfo(new ITXCRequestVideoInfoCallback() {
QOverride
public void onError(int errCode, String msg) {
Log.d(TAG, "onError msg = " + msg);
runOnUiThread(new Runnable() {
QOverride
public void run() {
Toast.makeText(VideoActivity.this, "onError msg = " + msg, Toast.LENGTH_SHORT).sh
ow();
}
});
}
@Override
public void onSuccess() {
Log.d(TAG, "onSuccess");
TXCStreamingInfo streamingInfo = mPlayerAssistor.getStreamingInfo();
Log.d(TAG, "streamingInfo = " + streamingInfo);
runOnUiThread(new Runnable() {
QOverride
public void run() {
if (mPlayerAssistor.getStreamingInfo() != null) {
// Play back the video
mSuperPlayerView.play(mPlayerAssistor.getStreamingInfo().playUrl);
} else {
Toast.makeText (VideoActivity.this, "streamInfo = null", Toast.LENGTH_SHORT).show
();
}
}
});
```



} });

Terminate the player after use

```
TXCPlayerAdapter.destroy();
```

# SDK API Description

# Initializing TXCPlayerAdatper

This API is used to initialize Adapter each time.

#### API

```
TXCPlayerAdapter.init(String appId);
```

#### **Parameter description**

appld: Enter the appid (if a subapplication is used, enter the subappid ). |

# Terminating TXCPlayerAdatper

This API is used to terminate Adapter. It can be called after the program exits.

#### API

TXCPlayerAdapter.destroy();

# Creating the Player auxiliary class

An auxiliary class of the player can be used to get the playback fileId and process DRM encryption APIs.

#### API

```
ITXCPlayerAssistor playerAssistor = TXCPlayerAdapter.createPlayerAssistor(String
fileId, String pSign);
```

### Parameter description

| Parameter | Туре   | Description                                 |
|-----------|--------|---|
| fileId    | String | The fileId of the video to be played. back. |



| Parameter | Туре   | Description           |
|-----------|--------|-----------------------|
| pSign     | String | The player signature. |

# Terminating the Player auxiliary class

This API is used to terminate an auxiliary class. You can call it when exiting the player or switching to the next video for playback.

#### API

TXCPlayerAdapter.destroyPlayerAssistor(ITXCPlayerAssistor assistor);

# **Requesting video playback information**

This API is used to request the stream information of the video to be played back from the Tencent Cloud VOD server.

#### API

playerAssistor.requestVideoInfo(ITXCRequestVideoInfoCallback callback);

#### Parameter description

| Parameter | Туре                         | Description              |
|-----------|------------------------------|--------------------------|
| callback  | ITXCRequestVideoInfoCallback | Async callback function. |

# Getting the basic video information

This API is used to get the video information and will take effect only after

playerAssistor.requestPlayInfo is called back.

# API

TXCVideoBasicInfo playerAssistor.getVideoBasicInfo();

# Parameter description

The parameters of TXCVideoBasicInfo are as follows:

| Parameter | Туре   | Description                    |
|-----------|--------|--------------------------------|
| name      | String | The video name.                |
| duration  | Float  | The video duration in seconds. |

| Parameter   | Туре   | Description            |
|-------------|--------|------------------------|
| description | String | The video description. |
| coverUrl    | String | The video thumbnail.   |

# Getting video stream information

This API is used to get the video stream information list and will take effect only after

```
playerAssistor.requestPlayInfo is called back.
```

#### API

TXCStreamingInfo playerAssistor.getStreamimgInfo();

#### Parameter description

#### TXCStreamingInfo

| Parameter  | Туре   | Description   |
|------------|--------|---|
| playUrl    | String | The playback URL.   |
| subStreams | List   | The adaptive bitrate substream information of the SubStreamInfo type. |

#### The parameters of SubStreamInfo are as follows:

| Parameter      | Туре   | Description  |
|----------------|--------|--|
| type           | String | Substream type. Valid values: video                                    |
| width          | Int    | The substream video width in px.                                       |
| height         | Int    | The substream video height in px.                                      |
| resolutionName | String | The specification name of the substream video displayed in the player. |

# Getting keyframe timestamp information

This API is used to get the video keyframe timestamp information and will take effect only after

playerAssistor.requestPlayInfo is called back.

#### API

List<TXCKeyFrameDescInfo> playerAssistor.getKeyFrameDescInfo();

#### **Parameter description**

The parameters of TXCKeyFrameDescInfo are as follows:

| Parameter  | Туре   | Description     |
|------------|--------|-----------------|
| timeOffset | Float  | 1.1             |
| content    | String | "Beginning now" |

# Getting thumbnail information

This API is used to get the thumbnail information and will take effect only after

```
playerAssistor.requestPlayInfo is called back.
```

#### API

TXCImageSpriteInfo playerAssistor.getImageSpriteInfo();

#### Parameter description

The parameters of TCXImageSpriteInfo are as follows:

| Parameter | Туре   | Description                                      |
|-----------|--------|--|
| imageUrls | List   | Array of thumbnail download URLs of String type. |
| webVttUrl | String | Thumbnail VTT file download URL.                 |

# Player Adapter for Web

Last updated : 2022-10-17 11:42:21

This document describes the Player Adapter for web, which helps you quickly integrate third-party players with VOD capabilities through flexible APIs to implement video playback. It supports the acquisition of basic video information, video stream information, and keyframe and thumbnail information as well as private encryption. This document is intended for developers with a basic knowledge of JavaScript.

# **SDK** Integration

Player Adapter for web supports integration via CDN and npm.

#### Integration via CDN

Import the initialization script to the page where videos need to be played back. The script will expose TcAdapter variables globally.

```
<script src="https://cloudcache.tencentcs.com/qcloud/video/dist/tcadapter.1.0.0.m
in.js"></script>
```

#### Integration via npm

```
// npm install
npm install tcadapter --save
// import TcAdapter
import TcAdapter from 'tcadapter';
```

# Placing Player Container

Add the container to the page where the player needs to be displayed. TcAdapter only needs to carry the container for video playback. The playback styles and custom features can be implemented by third-party players or yourself.

```
<video id="player-container-id">
</video>
```

# SDK Use Instructions

# Checking the development environment

Check whether the current environment supports TcAdapter.

```
TcAdapter.isSupported();
```

# **Initializing Adapter**

This API is used to initialize Adapter and create an Adapter instance. The initialization process will request the video file information from the Tencent Cloud VOD server.

#### API

```
const adapter = new TcAdapter('player-container-id', {
  fileID: string,
  appID: string,
  psign: string,
  hlsConfig: {}
 }, callback);
```

#### Parameter description

| Parameter | Туре              | Description  |
|-----------|-------------------|--|
| appID     | String            | The APPID of the VOD account.  |
| fileID    | String            | The fileId of the video to be played back.   |
| psign     | String            | The player signature.  |
| hlsConfig | HIsConfig         | HLS settings. Any parameters supported by hls.js can be used.  |
| callback  | TcAdapterCallBack | Callback for initialization completion. Basic video information can be obtained after this method is used. |

Note :

The underlying layer of TcAdapter is implemented based on hls.js, and it can receive any parameters supported by hls.js through HlsConfig to fine-tune playback.

# Getting the basic video information

This API is used to get the video information and will take effect only after initialization.

VideoBasicInfo adapter.getVideoBasicInfo();

#### Parameter description

The parameters of VideoBasicInfo are as follows:

| Parameter   | Туре   | Description                |
|-------------|--------|----------------------------|
| name        | String | The video name.            |
| duration    | Float  | Video duration in seconds. |
| description | String | The video description.     |
| coverUrl    | String | The video thumbnail.       |

# Getting video stream information

#### API

List<StreamingOutput> adapter.getStreamimgOutputList();

#### **Parameter description**

The parameters of StreamingOutput are as follows:

| Parameter  | Туре   | Description   |
|------------|--------|---|
| drmType    | String | The adaptive bitstream protection type. Valid values: plain (no encryption), simpleAES (HLS common encryption). |
| playUrl    | String | Playback URL  |
| subStreams | List   | The adaptive bitrate substream information of the SubStreamInfo type.   |

#### The parameters of SubStreamInfo are as follows:

| Parameter      | Туре   | Description  |
|----------------|--------|--|
| type           | String | Substream type. Valid values: video                      |
| width          | Int    | The substream video width in px.                         |
| height         | Int    | The substream video height in px.                        |
| resolutionName | String | The name of the substream video displayed in the player. |



# Getting keyframe timestamp information

### API

List<KeyFrameDescInfo> adapter.getKeyFrameDescInfo();

# Parameter description

The parameters of KeyFrameDescInfo are as follows:

| Parameter  | Туре   | Description     |
|------------|--------|-----------------|
| timeOffset | Float  | 1.1             |
| content    | String | "Beginning now" |

# Getting thumbnail information

#### API

ImageSpriteInfo adapter.getImageSpriteInfo();

#### Parameter description

The parameters of ImageSpriteInfo are as follows:

| Parameter | Туре   | Description                                      |
|-----------|--------|--|
| imageUrls | List   | Array of thumbnail download URLs of String type. |
| webVttUrl | String | Thumbnail VTT file download URL.                 |

# Listening on events

The player can perform event listening through the objects returned by the initialization, for example:

```
const adapter = TcAdapter('player-container-id', options);
adapter.on(TcAdapter.TcAdapterEvents.Error, function(error) {
   // do something
});
```

Here, type is the event type. Supported events include HLS' native events and the following events. Event names can be accessed from TcAdapter.TcAdapterEvents :

Name

Description



| Name           | Description   |
|----------------|---|
| LOADEDMETADATA | The corresponding video information was obtained through <code>playcgi</code> . Relevant video information can be obtained through this event callback. |
| HLSREADY       | An HLS instance was created. At this point, you can call the various attributes and methods of the HLS instance object.                                 |
| ERROR          | This event will be triggered when an error occurs. You can view the specific failure cause according to the callback parameters.                        |

#### Getting an HLS instance

The underlying layer of Adapter is implemented based on HLS.js. You can access an HLS instance and its attributes and methods through an Adapter instance to achieve fine control over the playback process.

```
adapter.on('hlsready', () => {
const hls = adapter.hls;
// ...
})
```

Note : For more information, see hls.js.

# Samples

#### Sample 1. Using TcAdapter in React

For the specific sample, see tcplayer/tcadapter-combine-video.

```
import { useEffect, useRef } from 'react';
import TcAdapter from 'tcadapter';
function App() {
if (!TcAdapter.isSupported()) {
throw new Error('current environment can not support TcAdapter');
}
const videoRef = useRef(null);
useEffect(() => {
const adapter = new TcAdapter(videoRef.current, {
appID: '1500002611',
fileID: '5285890813738446783',
```

```
psign: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDAwMjYxMSwiZmlsZUlk
IjoiNTI4NTq5MDqxMzczODQ0Njc4MyIsImN1cnJlbnRUaW11U3RhbXAiOjE2MTU5NTEyMzksImV4cGlyZ
VRpbWVTdGFtcCI6MjIxNTY1MzYyMywicGNmZyI6ImJhc2ljRHJtUHJlc2V0IiwidXJsQWNjZXNzSW5mby
I6eyJ0IjoiMjIxNTY1MzYyMyJ9fQ.hRrQYvC0UYtcO-ozB35k7LZI6E3ruvow7DC0XzzdYKE',
hlsConfig: {},
, () => \{
console.log('basicInfo', adapter.getVideoBasicInfo());
});
adapter.on(TcAdapter.TcAdapterEvents.HLSREADY, () => {
const hls = adapter.hls;
// ...
})
}, []);
const play = () => {
videoRef.current.play();
}
return (
<div>
<div>
<video id="player" ref={ videoRef }></video>
</div>
<button onClick={play}>play</button>
</div>
);
}
export default App;
```

# Sample 2. Combining TcAdapter with Video.js

For the specific sample, see tcplayer/tcadapter-combine-videojs.

```
// 1. Video.js playback over HLS uses `@videojs/http-streaming`, so we have devel
oped a set of policies for playback with TcAdapter to override the original logic
(you can also directly modify the internal logic of `@videojs/http-streaming`)
// src/js/index.js
import videojs from './video';
import '@videojs/http-streaming';
import './tech/tcadapter'; // Add logic
export default videojs;
// src/js/tech/tcadapter.js
import videojs from '../video.js';
import TcAdapter from 'tcadapter';
class Adapter {
    constructor(source, tech, options) {
        const el = tech.el();
    }
}
```
```
// Get parameters and initialize the instance
const adapter = new TcAdapter(el, {
appID: '1500002611',
fileID: '5285890813738446783',
psign: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDAwMjYxMSwiZmlsZUlk
IjoiNTI4NTg5MDgxMzczODQ0Njc4MyIsImN1cnJlbnRUaW11U3RhbXAiOjE2MTU5NTEyMzksImV4cGlyZ
VRpbWVTdGFtcCI6MjIxNTY1MzYyMywicGNmZyI6ImJhc2ljRHJtUHJlc2V0IiwidXJsQWNjZXNzSW5mby
I6eyJ0IjoiMjIxNTY1MzYyMyJ9fQ.hRrQYvC0UYtcO-ozB35k7LZI6E3ruvow7DC0XzzdYKE',
hlsConfig: {},
});
adapter.on(TcAdapter.TcAdapterEvents.LEVEL_LOADED, this.onLevelLoaded.bind(this
));
}
dispose() {
this.hls.destroy();
}
onLevelLoaded(event) {
this._duration = event.data.details.live ? Infinity : event.data.details.totaldur
ation;
}
}
let hlsTypeRE = /^application\/(x-mpegURL|vnd\.apple\.mpegURL)$/i;
let hlsExtRE = /\.m3u8/i;
let HlsSourceHandler = {
name: 'hlsSourceHandler',
canHandleSource: function (source) {
// skip hls fairplay, need to use Safari resolve it.
if (source.skipHlsJs || (source.keySystems && source.keySystems['com.apple.fps.1_
0'])) {
return '';
} else if (hlsTypeRE.test(source.type)) {
return 'probably';
} else if (hlsExtRE.test(source.src)) {
return 'maybe';
} else {
return '';
}
},
handleSource: function (source, tech, options) {
if (tech.hlsProvider) {
tech.hlsProvider.dispose();
tech.hlsProvider = null;
} else {
// After automatic loading is disabled for HLS, you need to manually load resourc
es
if (options.hlsConfig && options.hlsConfig.autoStartLoad === false) {
tech.on('play', function () {
```

```
if (!this.player().hasStarted()) {
this.hlsProvider.hls.startLoad();
}
});
}
}
tech.hlsProvider = new Adapter(source, tech, options);
return tech.hlsProvider;
},
canPlayType: function (type) {
if (hlsTypeRE.test(type)) {
return 'probably';
}
return '';
}
};
function mountHlsProvider(enforce) {
if (TcAdapter && TcAdapter.isSupported() || !!enforce) {
try {
let html5Tech = videojs.getTech && videojs.getTech('Html5');
if (html5Tech) {
html5Tech.registerSourceHandler(HlsSourceHandler, 0);
}
} catch (e) {
console.error('hls.js init failed');
}
} else {
// TcAdapter is not imported, MSE is not available, or X5 kernel is disabled
}
}
mountHlsProvider();
export default Adapter;
```

# Player SDK Policy Privacy Policy

Last updated : 2023-11-30 16:10:20

### 1. INTRODUCTION

This Module applies if you use Player SDK ("**Feature**"). This Module is incorporated into the privacy policy located at Privacy Policy. Terms used but not defined in this Module shall have the meaning given to them in the Privacy Policy. In the event of any conflict between the Privacy Policy and this Module, this Module shall apply to the extent of the inconsistency.

#### 2. CONTROLLERSHIP

The controller of the personal information described in this Module is as specified in the Privacy Policy.

### 3. AVAILABILITY

This Feature is available to users globally but primarily intended for users located in the same country/region as the selected service region for optimal performance.

#### 4. HOW WE USE PERSONAL INFORMATION

We will use the information in the following ways and in accordance with the following legal basis:

| Personal Information | Use | Legal Basis |
|----------------------|-----|-------------|
|----------------------|-----|-------------|



| Personal Information  | Use   | Legal Basis   |
|---|---|---|
| <ul> <li>Operation Data: configuration information of customer's<br/>Tencent Cloud console, information about the customer's<br/>network fluctuations, audio and video quality problems, SDK<br/>logs relating to the use of the SDK, customer SDK version<br/>number and OS type</li> <li>Log Data: log files of the customer's backend, e.g.<br/>network download log, video decoding and encoding log,<br/>rendering screen log, API interface call log</li> </ul> | We use this<br>information for<br>troubleshooting,<br>operation and<br>maintenance<br>analysis.<br>Please note that<br>this data is stored<br>and backed up in<br>TencentDB for<br>MySQL<br>("MySQL").                                  | We process this<br>information as it is<br>necessary for us to<br>perform our contract<br>with you to provide the<br>Feature. |
| End User Network and Device Information: WiFi status, system properties, device model, operating system, sensor information, user IP address and user agent information.  | We use this<br>information for:<br>• playback<br>quality analysis;<br>and<br>•<br>troubleshooting,<br>operation and<br>maintenance<br>analysis.<br>Please note that<br>this data is stored<br>and backed up in<br>our MySQL<br>feature. | We process this<br>information as it is<br>necessary for us to<br>perform our contract<br>with you to provide the<br>Feature. |

| Personal Information  | Use  | Legal Basis   |
|---|--|---|
| Administrative Data: customer's app name, application<br>package name (identifier for listing app on application store),<br>bundle ID (identifier for listing app on application store),<br>appID.<br>Customer License Data: license type, license url and<br>license key (to allow you to unlock access to the SDK),<br>license details (expiry date, function module, automatic<br>renewal or not). | We use this<br>information to:<br>• provide and<br>allow you to<br>access the<br>Feature; and<br>• for<br>troubleshooting<br>analysis.<br>Please note that<br>this data is stored<br>and backed up in<br>our MySQL<br>feature. | We process this<br>information as it is<br>necessary for us to<br>perform our contract<br>with you to provide the<br>Feature. |

## 5. HOW WE SHARE AND STORE PERSONAL INFORMATION

As specified in the Privacy Policy.

## 6. DATA RETENTION

We will retain personal information in accordance with the following:

| Personal<br>Information                | Retention Policy   |
|--|--|
| Operational<br>data                    | Log data: We retain such data for 14 days.   |
|  | Operational data other than log data: We retain such data for as long as you use the Feature.<br>When your use of the Feature is terminated, we will delete this data within 5 days.   |
| End user<br>network and<br>device data | Automatically deleted after 14 days or for a longer or shorter retention period as requested by you. Under special circumstances, if the user account is attacked or hacked, we will delete the data.  |
| Administrative<br>data                 | This data will be deleted within 5 days of your request to deactivate your Tencent Cloud account, or immediately upon your data deletion request. Please note that if you submit a data deletion request, we will not be able to continue to provide you with the Feature. |