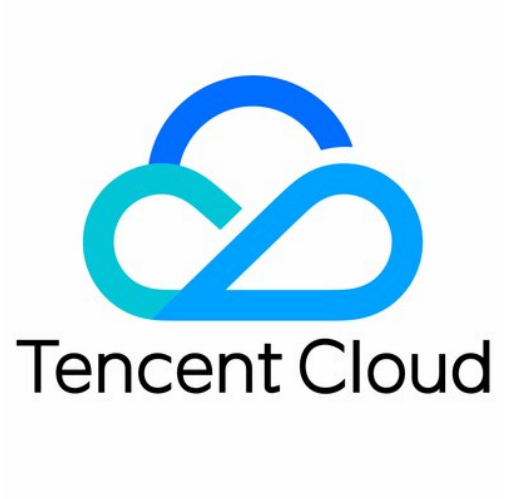
Mobile Live Video Broadcasting

Playback

Product Documentation



Contents

Playback
iOS
LVB

Last updated：2024-01-13 15:49:41

Basics

This document introduces the live playback feature of the Video Cloud SDK.

Live streaming and video on demand

In **live streaming**, the video streams published by hosts in real time are the source of streaming. When hosts stop publishing streams, the video played stops. Since video is streamed in real time, players do not have progress bars when they play live streaming URLs.
In **video on demand (VOD)**, video files in the cloud are the source of streaming. Videos can be played at any time as long as they are not deleted from the cloud, and the playback progress can be adjusted using the progress bar. Video streaming websites such as Tencent Video and Youku Tudou are typical applications of VOD.

Supported protocols

The table below lists the common protocols used for live streaming. We recommend FLV URLs (which start with `http` and end with `flv`) for LVB and WebRTC for LEB. For more information, please see LEB.

Protocol	Pro	Con	Playback Latency
HLS	Mature, well adapted to high-concurrency scenarios	SDK integration is required.	3s - 5s
FLV	Mature, well adapted to high-concurrency scenarios	SDK integration is required	2s - 3s
RTMP	Relatively low latency	Poor performance in high-concurrency scenarios	1s - 3s
WebRTC	Lowest latency	SDK integration is required	< 1s

Note:
LVB and LEB are priced differently. For details, please see LVB Billing Overview and LEB Billing Overview.

Notes

The Video Cloud SDK **does not impose any limit on the sources of playback URLs**, which means you can use it to play both Tencent Cloud and non-Tencent Cloud URLs. However, the player of the SDK supports only live streaming URLs in FLV, RTMP, HLS (M3U8), and WebRTC formats and VOD URLs in MP4, HLS (M3U8), and FLV formats.

Sample Code

Regarding frequently asked questions among developers, Tencent Cloud offers a straightforward API example project, which you can use to quickly learn how to use different APIs.

Platform	GitHub Address
iOS	Github
Android	Github
Flutter	Github

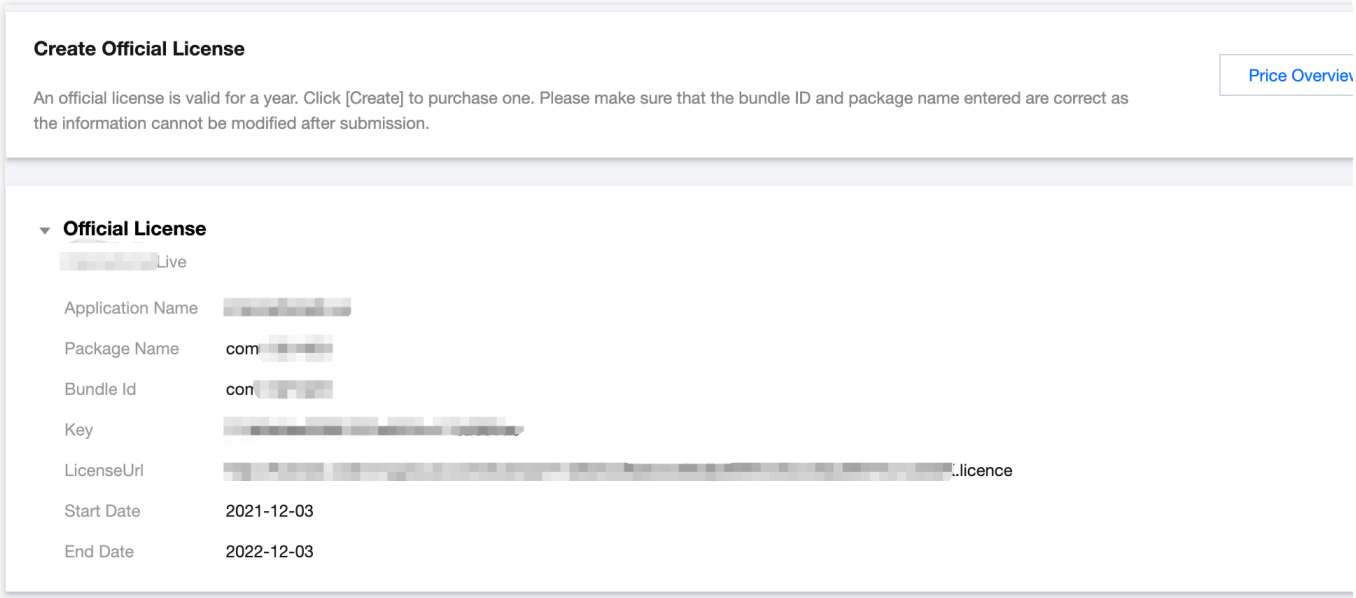
Integration

Step 1. Download the SDK

Download the SDK and follow the instructions in SDK Integration to integrate the SDK into your application.

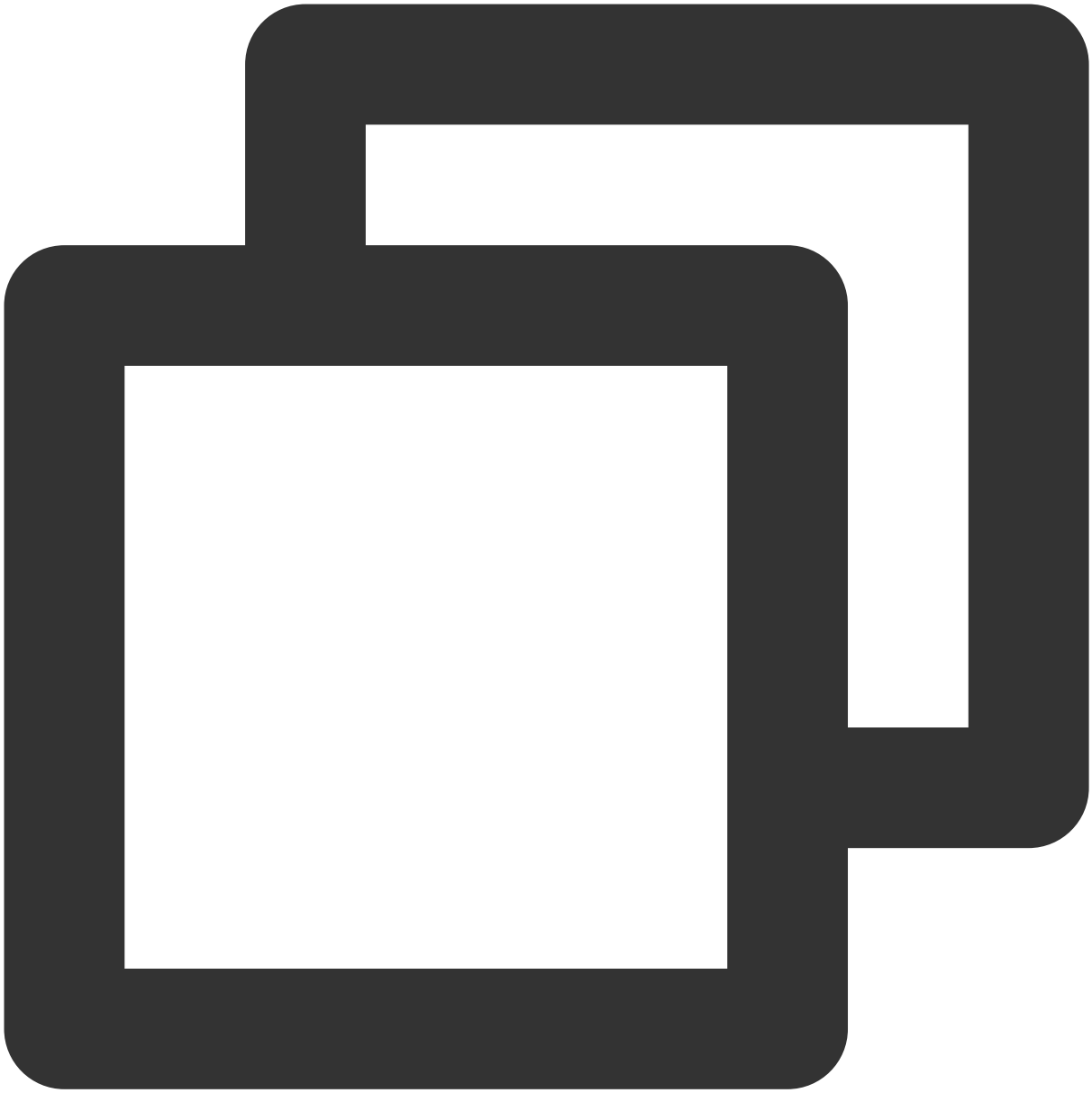
Step 2. Configure License Authorization for SDK

1. Obtain license authorization：
If you have obtained the relevant license authorization， need to Get License URL and License Key in Cloud Live Console



If you have not yet obtained the license authorization, Please reference Adding and Renewing Licenses to make an application.

2. Before your App calls SDK-related functions (it is recommended in the Application class), set the following settings:



```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    NSString * const licenceURL = @"<your licenseUrl>";
    NSString * const licenceKey = @"<your key>";

    [V2TXLivePremier setEnvironment:@"GDPR"];// set your environment
    // V2TXLivePremier located in "V2TXLivePremier.h"
    [V2TXLivePremier setLicence:licenceURL key:licenceKey];
    [V2TXLivePremier setObserver:self];
    NSLog(@"SDK Version = %@", [V2TXLivePremier getSDKVersionStr]);
    return YES;
}
```

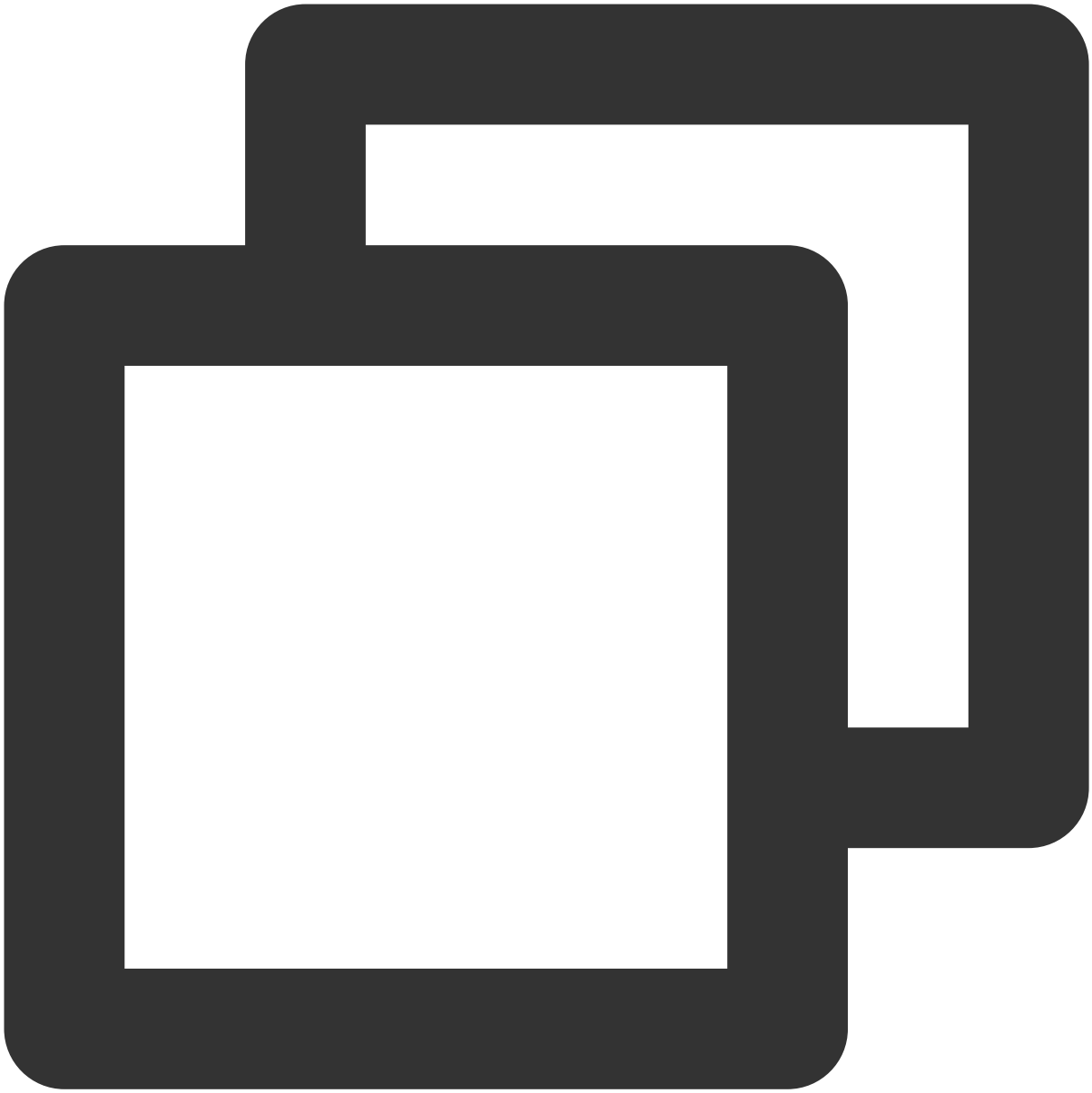
```
#pragma mark - V2TXLivePremierObserver
- (void)onLicenceLoaded:(int)result Reason:(NSString *)reason {
    NSLog(@"onLicenceLoaded: result:%d reason:%@", result, reason);
}
@end
```

Note:

The packageName configured in the license must be the same as the application itself, otherwise the play stream will fail.

Step 3. Create Player

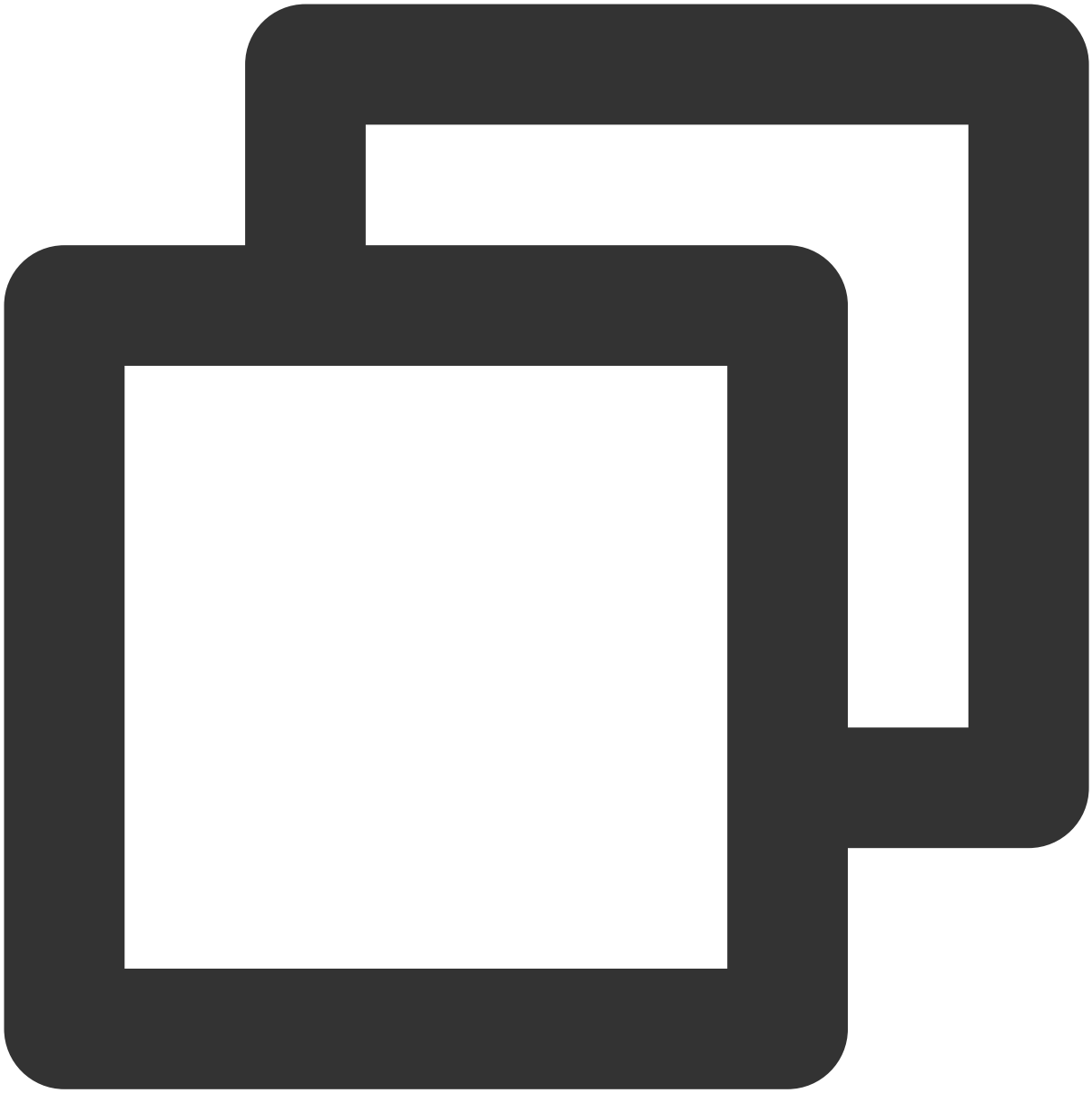
The V2TXLivePlayer module in Tencent Cloud SDK is responsible for implementing the live broadcast function.



```
V2TXLivePlayer *_txLivePlayer = [[V2TXLivePlayer alloc] init];
```

Step 4. Create a rendering view

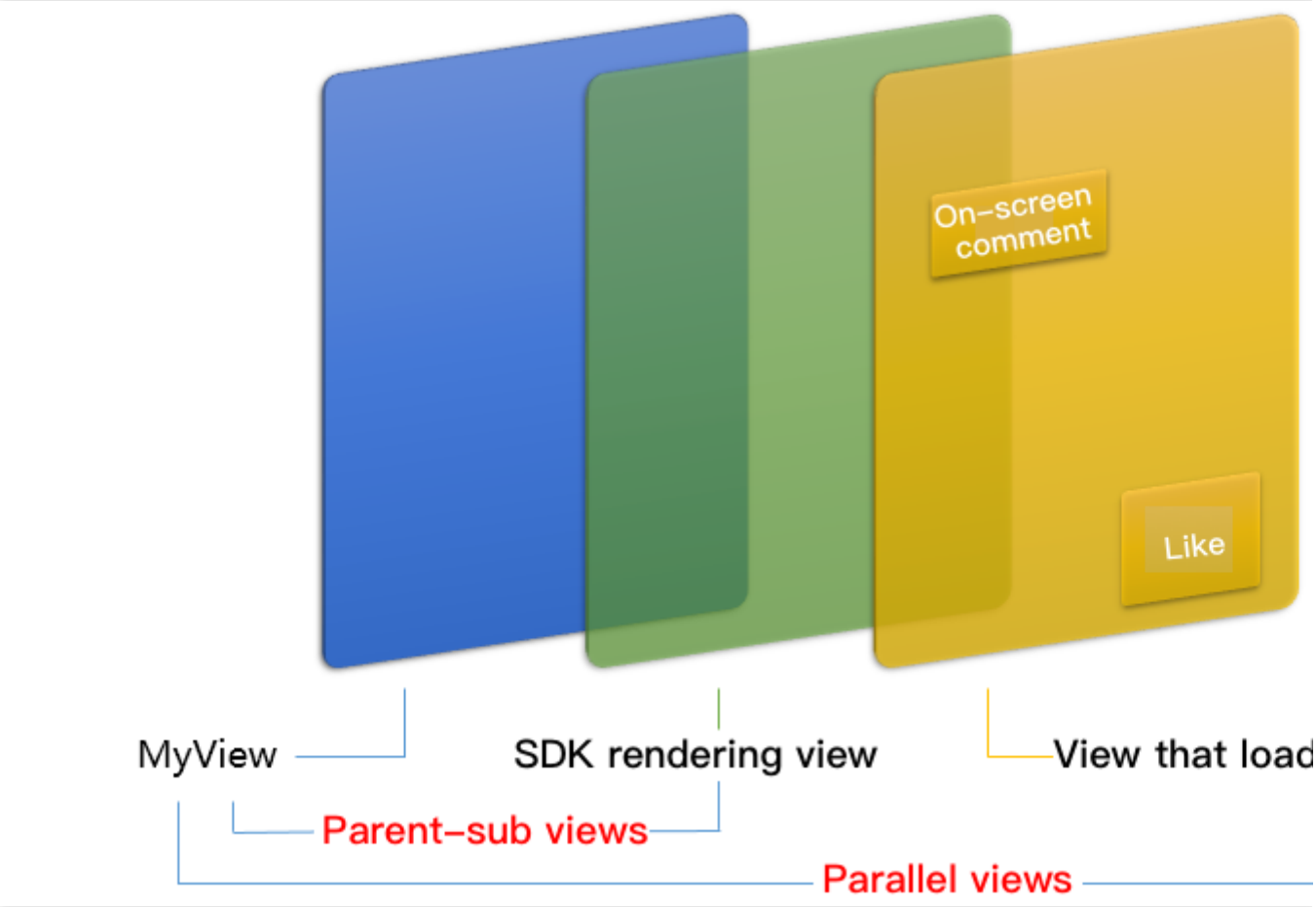
In iOS, a view is used as the basic UI rendering unit. Therefore, you need to configure a view, whose size and position you can adjust, for the player to display video images on.



```
// Use setRenderView to bind a rendering view to the player
[_txLivePlayer setRenderView:_myView];
```

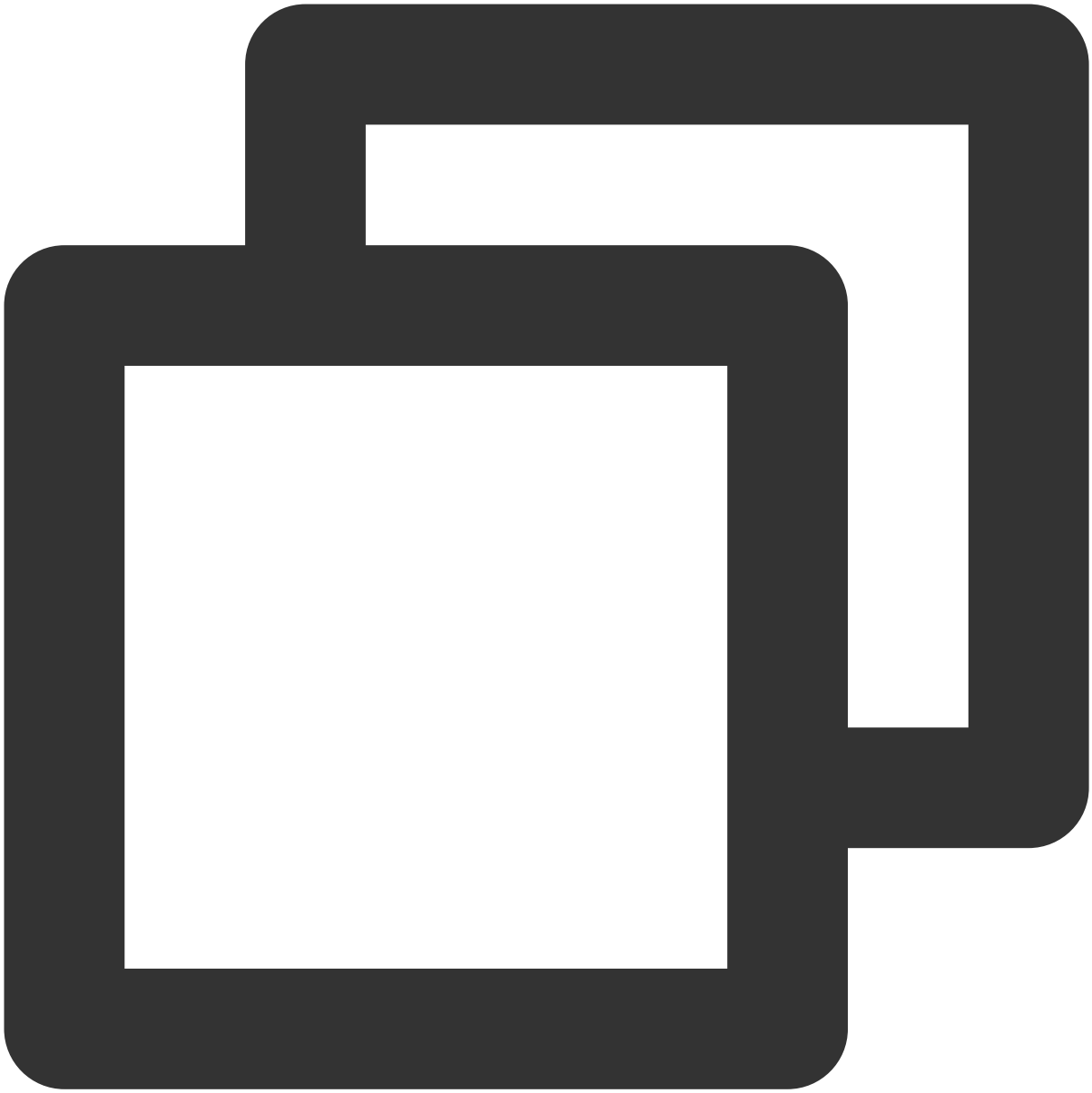
Technically, the player does not render video images directly on the view (`_myView` in the sample code) you provide. Instead, it creates a subview for OpenGL rendering over the view.

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.



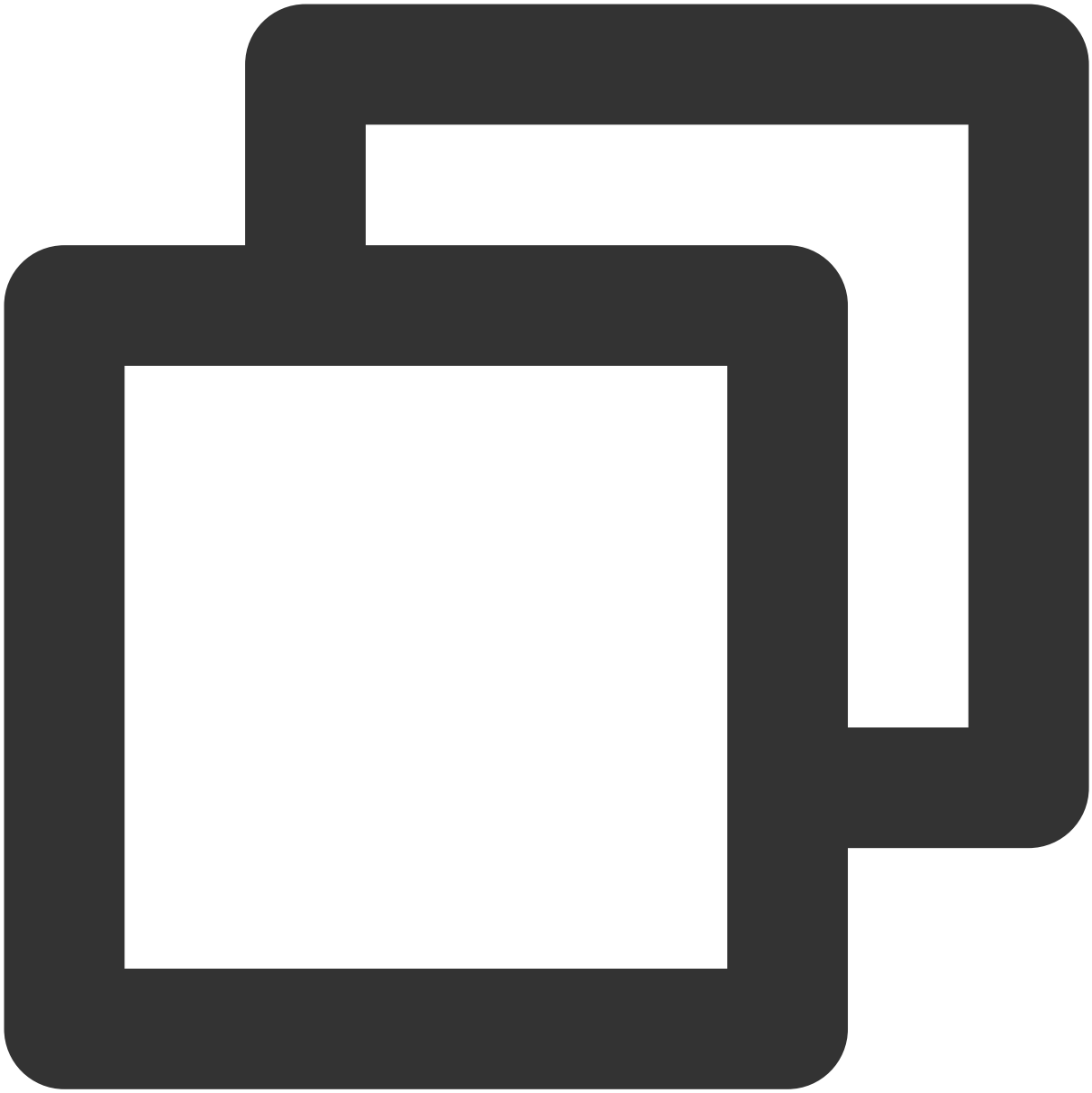
How can I animate views?

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



```
NSString* url = @"http://2157.liveplay.myqcloud.com/live/2157_xxxx.flv";
[_txLivePlayer startPlay:url];
```

Step 6. Change the fill mode

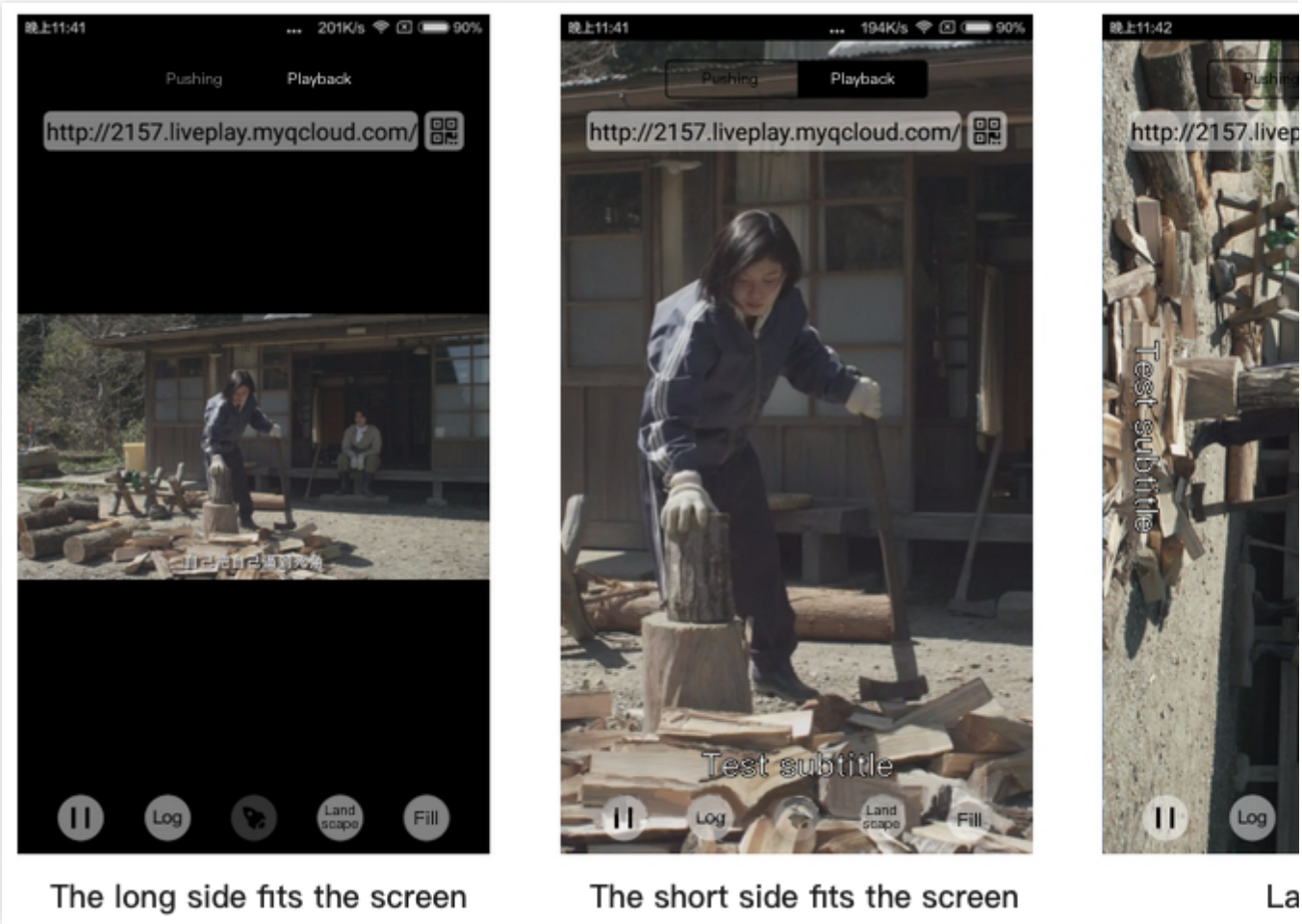
setRenderFillMode: aspect fill or aspect fit

Value	Description
V2TXLiveFillModeFill	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.

V2TXLiveFillModeFit	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: clockwise rotation of video

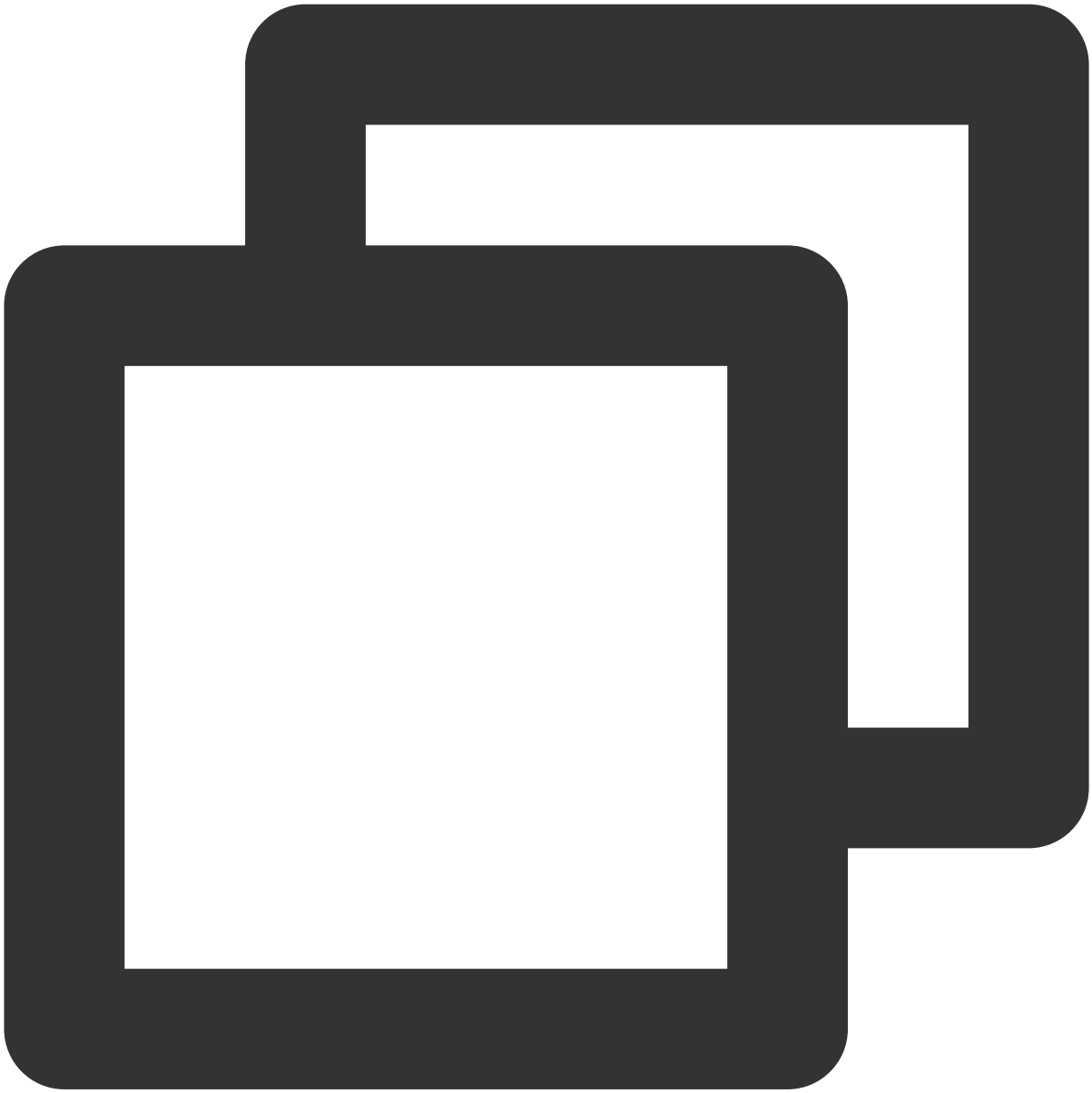
Value	Description
V2TXLiveRotation0	Original
V2TXLiveRotation90	Rotate 90 degrees clockwise
V2TXLiveRotation180	Rotate 180 degrees clockwise
V2TXLiveRotation270	Rotate 270 degrees clockwise



The long side fits the screen The short side fits the screen La

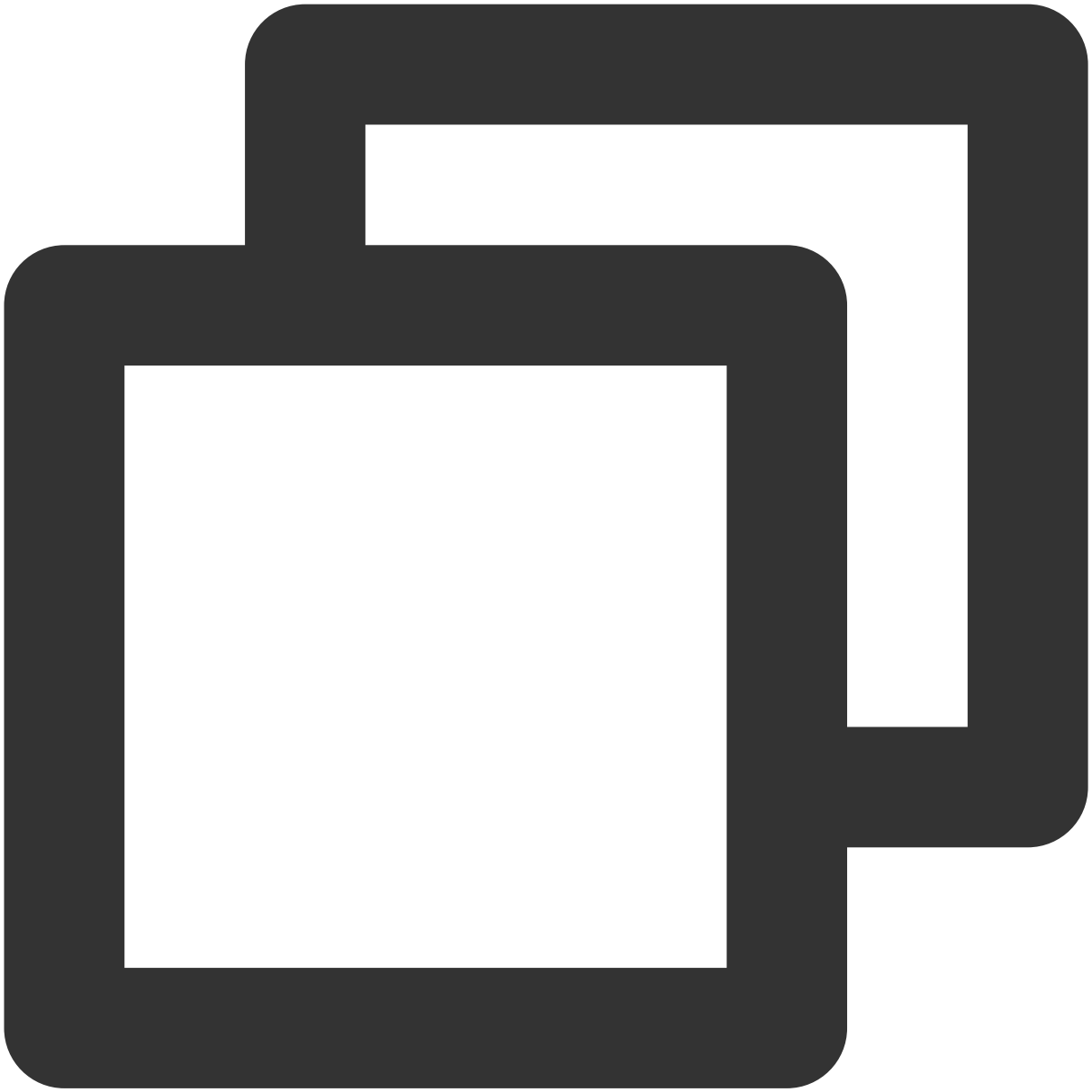
Step 7. Pause playback

Technically speaking, you cannot pause a live playback. In this document, by pausing playback, we mean **freezing video** and **disabling audio**. In the meantime, new video streams continue to be sent to the cloud. When you resume playback, it starts from the time of resumption. This is in contrast to VOD. With VOD, when you pause and resume playback, the player behaves the same way as it does when you pause and resume a local video file.



```
// Pause playback
[_txLivePlayer pauseAudio];
[_txLivePlayer pauseVideo];
// Resume playback
[_txLivePlayer resumeAudio];
[_txLivePlayer resumeVideo];
```

Step 8. Stop playback

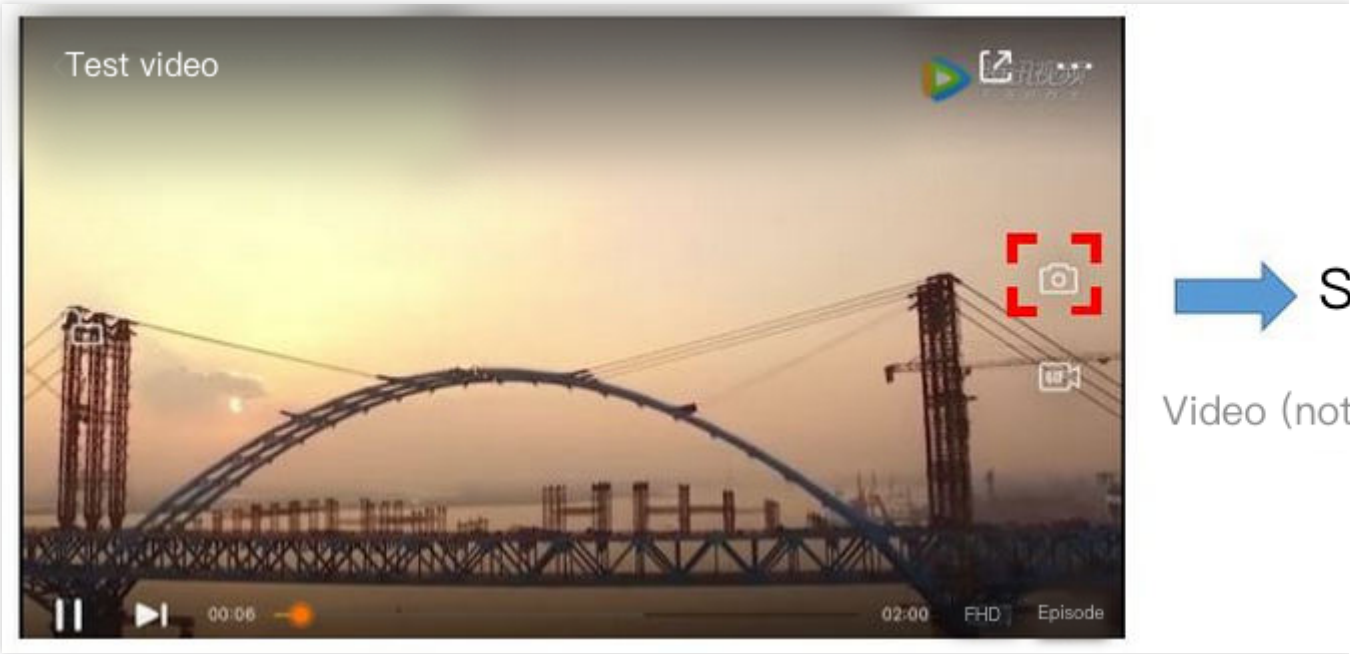


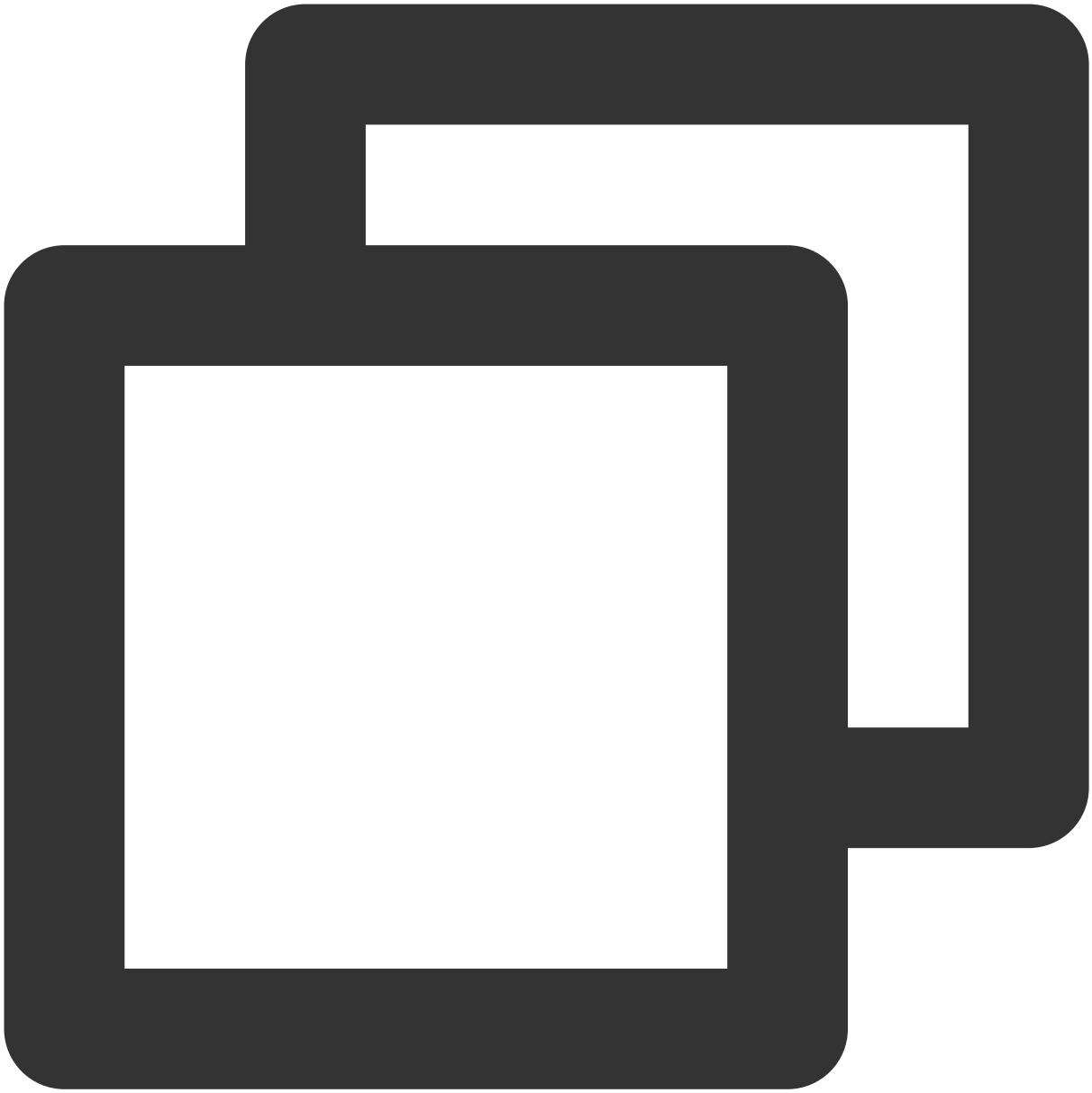
```
// Stop playback
[_txLivePlayer stopPlay];
```

Step 9. Take a screenshot

Call **snapshot** to take a screenshot of the live video streamed. You can get the screenshot taken in the onSnapshotComplete callback of `V2TXLivePlayerObserver`. This method captures a frame of the streamed

video. To capture the UI, use the corresponding API of the iOS system.





```
...
[_txLivePlayer setObserver:self];
[_txLivePlayer snapshot];
...

- (void)onSnapshotComplete:(id<V2TXLivePlayer>)player image:(TXImage *)image {
    if (image != nil) {
        dispatch_async(dispatch_get_main_queue(), ^{
            [self handle:image];
        });
    }
```

```
}
```

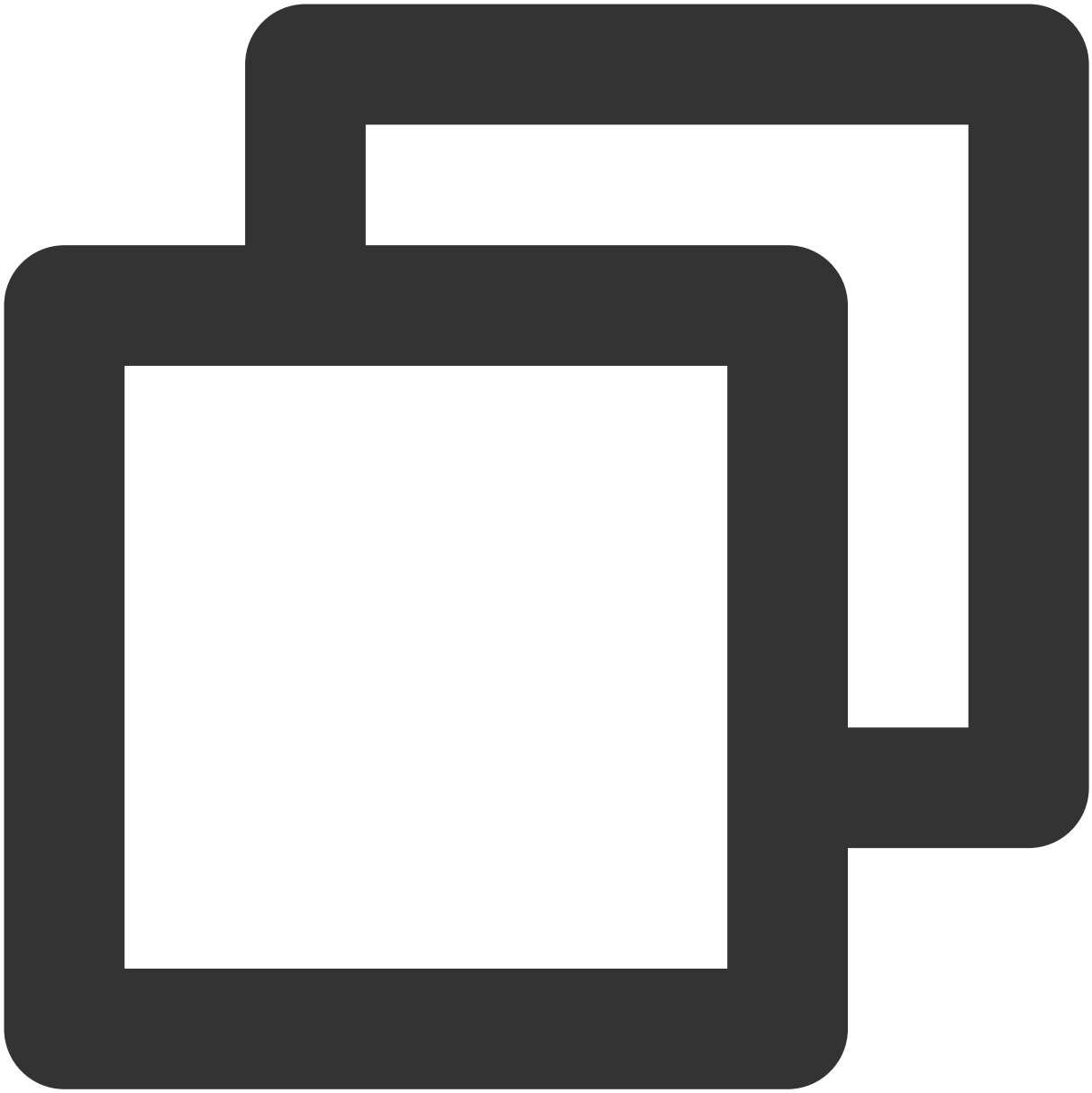
Latency Control

The live playback feature of the SDK is not based on FFmpeg, but Tencent Cloud's proprietary playback engine, which is why the SDK offers better latency control than open-source players do. We provide three latency control modes, which can be used for showrooms, game streaming, and hybrid scenarios.

Comparison of the three modes

Mode	Stutter	Average Latency	Scenario	Remarks
Speedy	More likely than the speedy mode	2-3s	Live showroom (Chongding Dahui)	The mode delivers low latency and is suitable for latency-sensitive scenarios.
Smooth	Least likely of the three	>= 5s	Game streaming (Penguin Esports)	Playback is least likely to stutter in this mode, which makes it suitable for ultra-high-bitrate streaming of games such as PUBG.
Auto	Self-adaptive to network conditions	2-8s	Hybrid	The better network conditions at the audience end, the lower the latency.

Code to integrate the three modes



```
// Auto mode
[_txLivePlayer setCacheParams:1 maxTime:5];
// Speedy mode
[_txLivePlayer setCacheParams:1 maxTime:1];
// Smooth mode
[_txLivePlayer setCacheParams:5 maxTime:5];

// Start playback after configuration
```

Note:

For more information on stuttering and latency control, see Video Stutter.

Listening for SDK Events

You can bind a V2TXLivePlayerObserver to your `V2TXLivePlayer` object to receive callback notifications about the player status, playback volume, first audio/video frame, statistics, warning and error messages, etc.

Periodically triggered notifications

The onStatisticsUpdate callback notification is triggered every 2 seconds to update you on the player's status in real time. Like a car's dashboard, the callback gives you information about the SDK, such as network conditions and video information.

Parameter	Description
appCpu	CPU usage (%) of the application
systemCpu	CPU usage (%) of the system
width	Video width
height	Video height
fps	Frame rate (fps)
audioBitrate	Audio bitrate (Kbps)
videoBitrate	Video bitrate (Kbps)

The onPlayoutVolumeUpdate callback, which notifies you of the player's volume, works only after you call enableVolumeEvaluation to enable the volume reminder. You can set the interval of the callback by specifying the `intervalMs` parameter of `enableVolumeEvaluation`.

Event-triggered notifications

Other callbacks are triggered when specific events occur.

LEB

Last updated：2024-01-13 15:49:41

LEB Overview

Live Event Broadcasting (LEB) is the ultra-low-latency version of LVB. It features lower latency than traditional streaming protocols and delivers superior playback experience with millisecond latency. It is suitable for scenarios with high requirements on latency, such as online education, sports streaming, and online quizzes.

Note:

The figure above (made using scrcpy) is a comparison of LEB and standard CDN live streaming. The images on the left and in the middle show the playback end of standard CDN live streaming and **LEB**, and the image on the right shows the publishing end.

LVB and LEB are priced differently. For details, please see LVB Billing Overview and LEB Billing Overview.

Strengths

Strength	Description
Playback with millisecond latency	The latency is kept within 1s thanks to the use of UDP, as opposed to 3-5s in traditional live streaming. This, along with excellent instant streaming performance and low stuttering rate, guarantees a superior streaming experience.
Diverse features and smooth migration	LEB integrates a wide range of features including stream publishing, transcoding, recording, screenshot, porn detection, and playback. It allows smooth migration from standard live streaming.
Easy-to-use, secure, and reliable	The use of a standard protocol makes integration easy. You can play live video on Chrome and Safari without installing any plugins. In addition, the playback protocol encrypts video by default for improved security and reliability.

Use cases

Scenario	Description
Sports event	LEB offers ultra-low-latency streaming for sports events. It brings sports content to audience at low latency, allowing audience to learn what's happening in real time.
E-commerce streaming	Some e-commerce streaming scenarios, for example, online auctions and

	sales promotion, require extremely low latency. LEB's ability to stream at ultra-low latency ensures that hosts and audience get real-time feedback from each other, improving online shopping experience.
Online classes	LEB can be used for online classes. Its ability to stream at ultra-low latency allows teachers and students to interact with each other as they do in offline classes.
Online quizzes	Due to latency, some online quizzes have to insert extra frames at the audience end to ensure that the host and audience are in sync with each other. This is not necessary if you use LEB, whose ultra-low-latency streaming capability makes sure that the two sides are in sync. It helps you implement online quizzes more easily and deliver smoother experience.
Showrooms	LEB can significantly improve the experience of latency-sensitive interactions such as gift giving in live showrooms.

Tryout

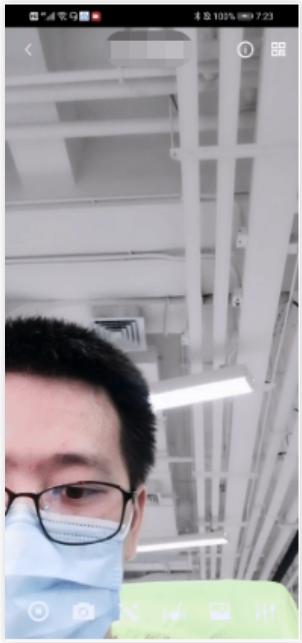
Video Cloud Toolkit is a comprehensive audio-video solution developed by Tencent Cloud that allows you to try out the features of the TRTC, MLVB and UGC SDKs, including the **LEB Player**.

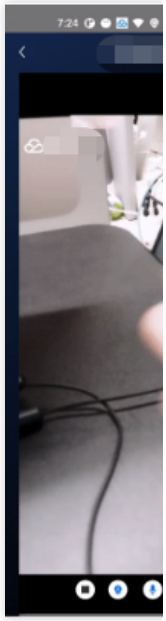
Note:

The demonstration and directions in this document use the demo app for Android as an example. The UI of the app for iOS is slightly different.

Source code and demonstration

Source Code	Demo	Publishing Demonstration (Android)	Playback Demonstr
Android			
iOS			





Note:

In addition to the above sample code, regarding frequently asked questions among developers, Tencent Cloud offers a straightforward API example project, which you can use to quickly learn how to use different APIs.

iOS：MLVB-API-Example

Android：MLVB-API-Example

Flutter：Live-API-Example

Publishing

LEB is compatible with LVB, which means you can publish streams using an ordinary publisher and play the streams using LEB.

1. Download and install Video Cloud Toolkit, log in, and go to **MLVB > Push (Camera)**.
2. Allow the permissions asked, and tap **Auto-generate** to start publishing streams.
3. If publishing is successful, tap the QR code icon in the top right and select **LEB** to get the playback URL for LEB.
4. During publishing, you can tap the menu icon in the bottom right to apply filters, add background music, switch cameras, etc.

Playback

1. Download and install Video Cloud Toolkit, log in, and go to **Live Broadcast > LEB Player**.
2. Allow the permissions asked, tap the scan button, and scan the LEB playback URL obtained earlier.
3. Playback starts automatically once the QR code is read. During playback, you can tap the menu icon in the bottom right to mute playback or change other settings.

Integration

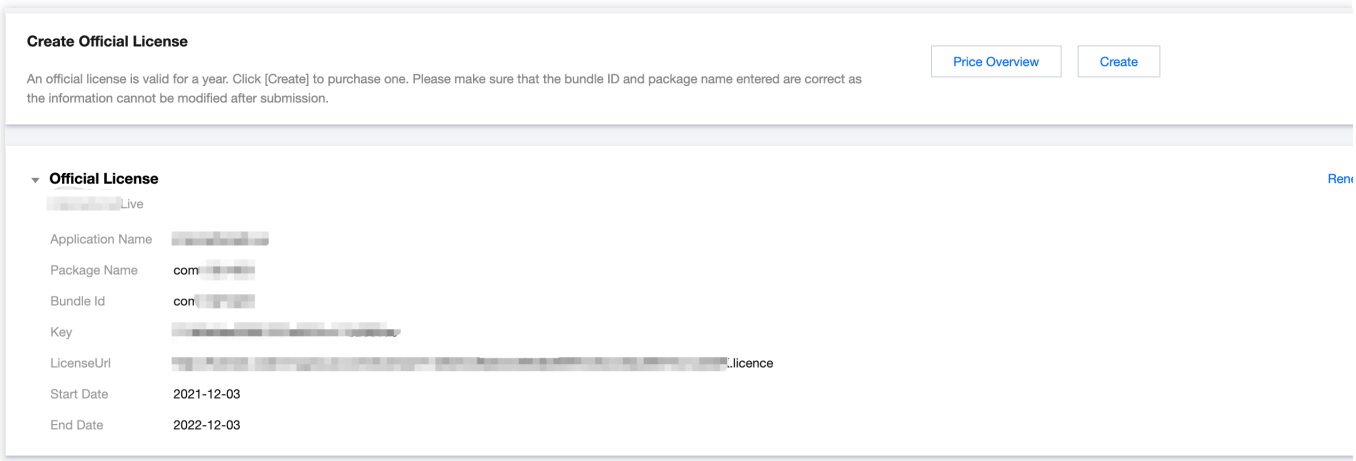
In the new version of the MLVB SDK, you can use V2TXLivePlayer for LEB and `V2TXLivePusher` for publishing. LEB supports WebRTC protocols and uses the standard extension method. All URLs in LEB start with `webrtc://` .

Step 1. Download the SDK

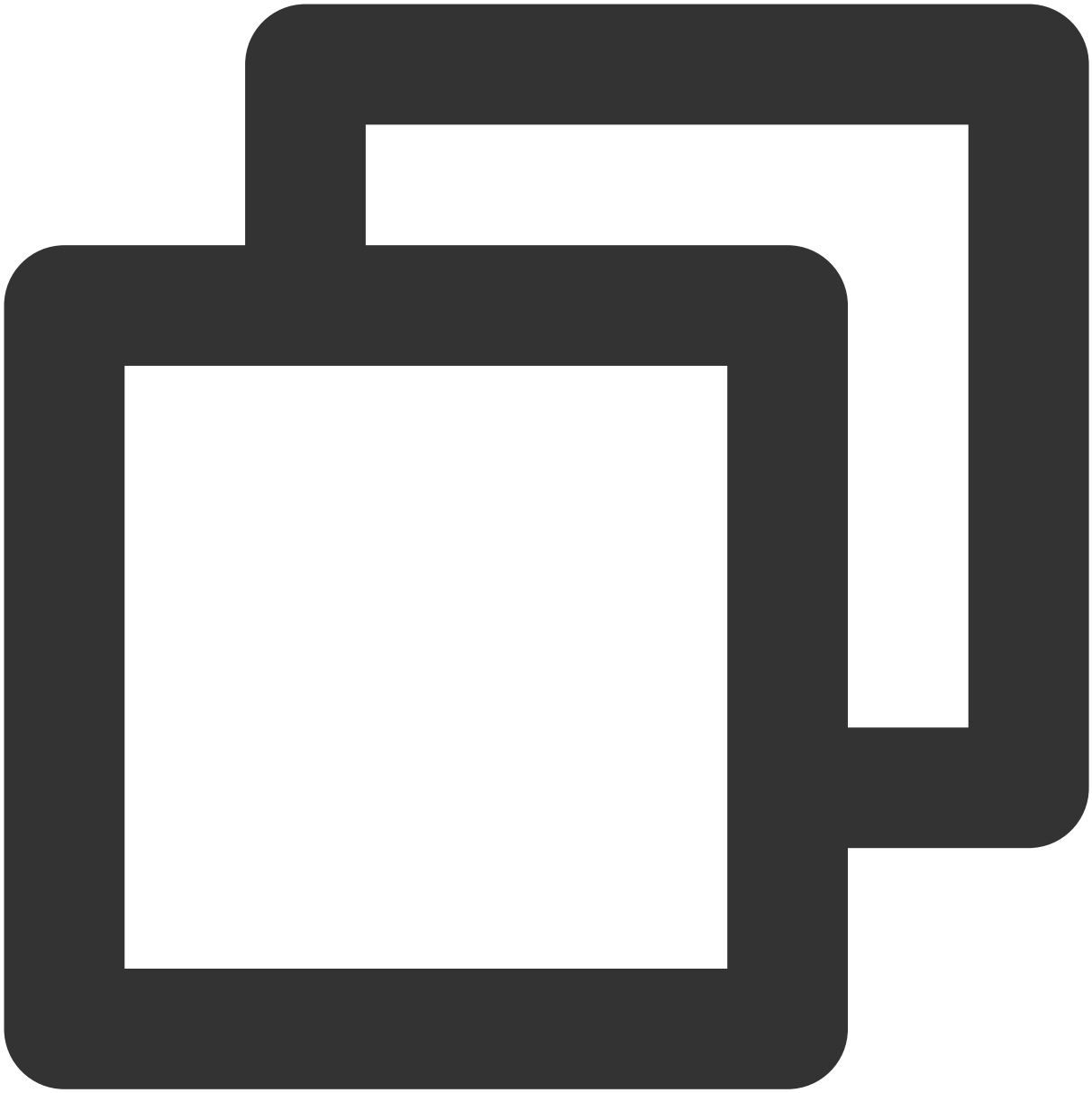
Download LiteAV_All or Professional at SDK Download.

Step 2. Configure License Authorization for SDK

1. Obtain license authorization：
If you have obtained the relevant license authorization， Need to Get License URL and License Key in Cloud Live Console



If you have not yet obtained the license authorization， Please reference Adding and Renewing Licenses to make an application.
2. Before your App calls SDK-related functions (it is recommended in the Application class), set the following settings:



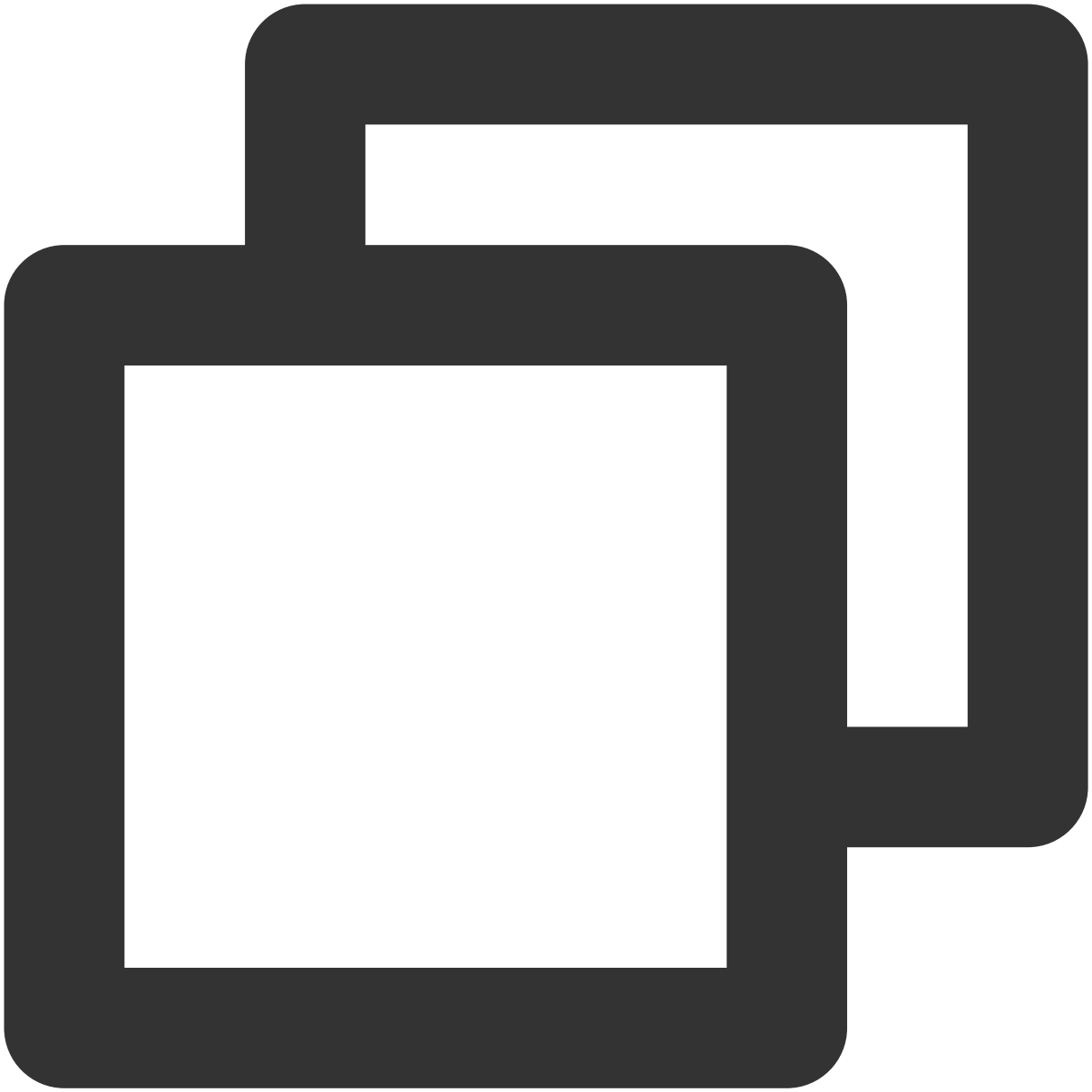
```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    NSString * const licenceURL = @"<your licenseUrl>";
    NSString * const licenceKey = @"<your key>";

    // V2TXLivePremier located in "V2TXLivePremier.h"
    [V2TXLivePremier setEnvironment:@"GDPR"];// set your environment
    [V2TXLivePremier setLicence:licenceURL key:licenceKey];
    [V2TXLivePremier setObserver:self];
    NSLog(@"SDK Version = %@", [V2TXLivePremier getSDKVersionStr]);
    return YES;
}
```

```
#pragma mark - V2TXLivePremierObserver
- (void)onLicenceLoaded:(int)result Reason:(NSString *)reason {
    NSLog(@"onLicenceLoaded: result:%d reason:%@", result, reason);
}
@end
```

Step 3. Get a playback URL

In live streaming, URLs are needed for both publishing and playback. For instructions on how to get URLs for LEB, please see Live Event Broadcasting (LEB) > Get Playback URL.

All LEB URLs start with `webrtc://` , as in:



```
webrtc://{Domain}/{AppName}/{StreamName}
```

The table below lists the key fields in an LEB URL and their meanings.

Field	Description
webrtc://	Prefix
Domain	Domain name
AppName	Application name, which is `live` by default. It specifies the storage path of a live

	streaming file.
StreamName	Stream name, which is the unique identifier of a stream

Note:

To publish streams, please see Publishing from Camera or Publishing from Screen.

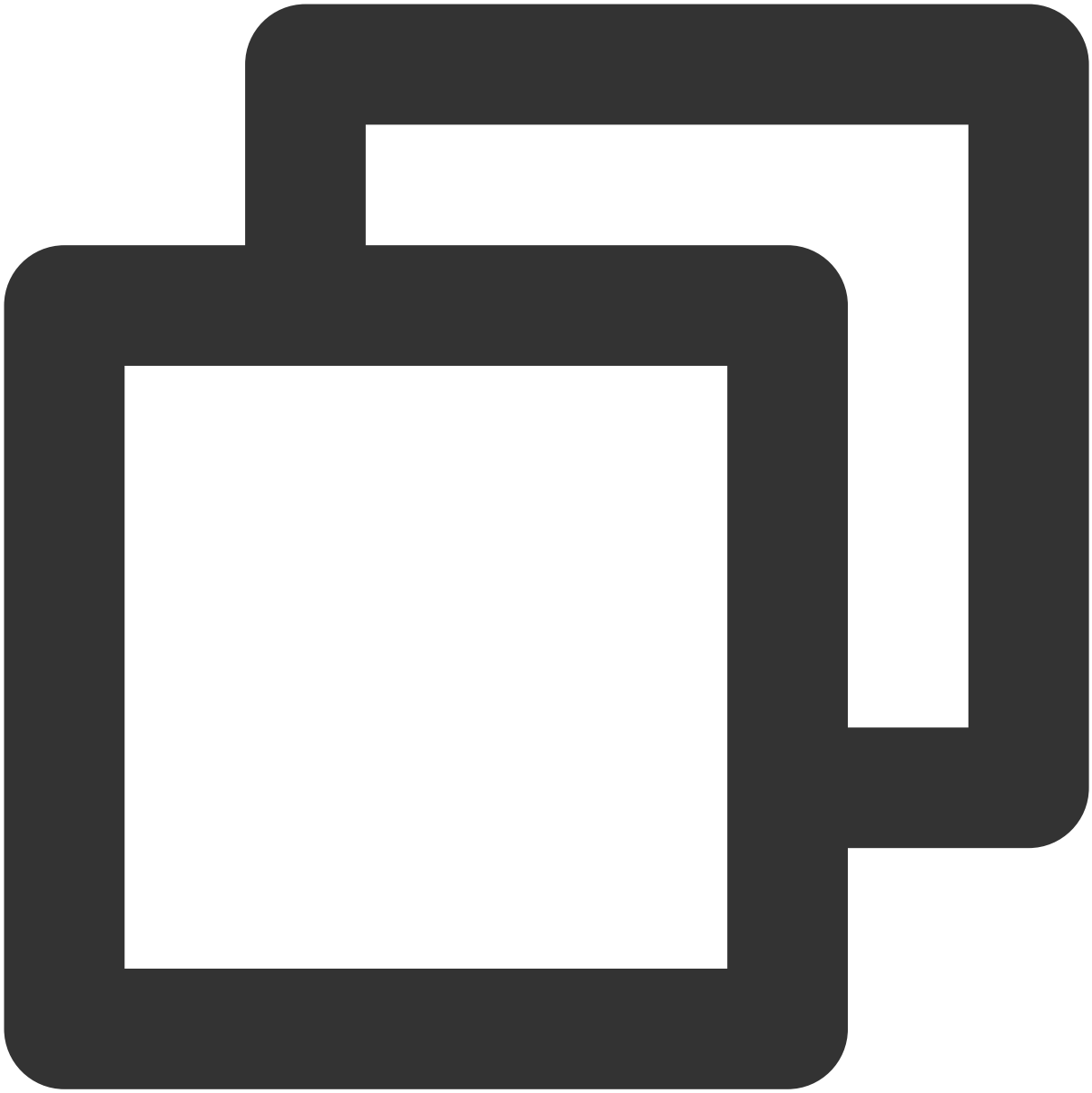
Step 4. Start LEB

You can use a `V2TXLivePlayer` object for LEB. For details, see the code below (make sure that you pass in the correct URL).

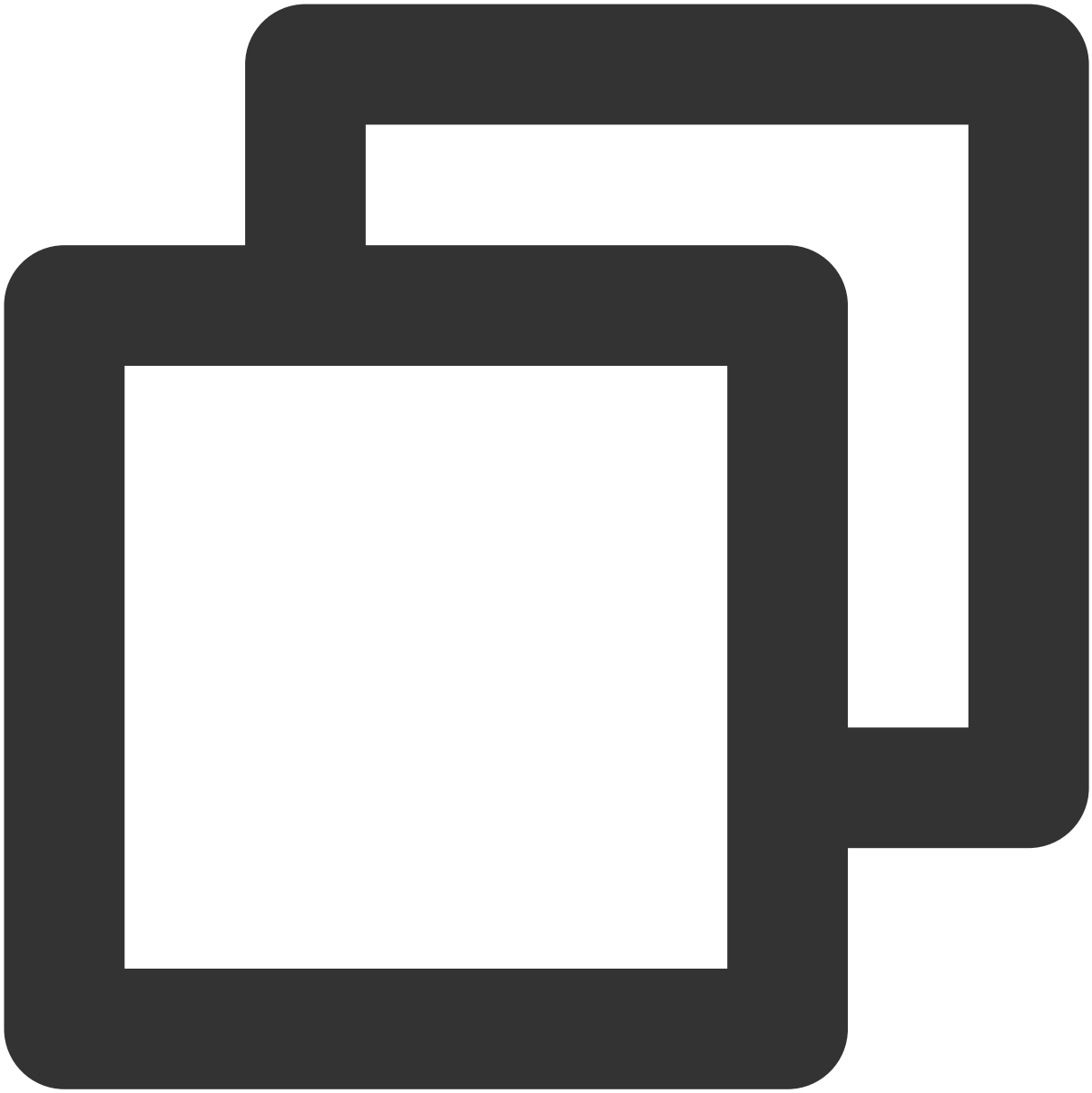
Sample code

Android

iOS



```
// Create a V2TXLivePlayer object
V2TXLivePlayer player = new V2TXLivePlayerImpl(mContext);
player.setObserver(new MyPlayerObserver(playerView));
player.setRenderView(mSurfaceView);
// Pass in the low-latency playback URL to start playback
player.startPlay("webrtc://{Domain}/{AppName}/{StreamName}");
```



```
V2TXLivePlayer *player = [[V2TXLivePlayer alloc] init];
[player setObserver:self];
[player setRenderView:videoView];
[player startPlay:@"webrtc://{Domain}/{AppName}/{StreamName}"];
```

Android
LVB

Last updated：2024-01-13 15:49:41

Basics

This document introduces the live playback feature of the Video Cloud SDK.

Live streaming and video on demand

In **live streaming**, the video streams published by hosts in real time are the source of streaming. When hosts stop publishing streams, the video played stops. Since video is streamed in real time, players do not have progress bars when they play live streaming URLs.
In **video on demand (VOD)**, video files in the cloud are the source of streaming. Videos can be played at any time as long as they are not deleted from the cloud, and the playback progress can be adjusted using the progress bar. Video streaming websites such as Tencent Video and Youku Tudou are typical applications of VOD.

Supported protocols

The table below lists the common protocols used for live streaming. We recommend FLV URLs (which start with `http` and end with `flv`) for LVB and WebRTC for LEB. For more information, please see LEB.

Protocol	Pro	Con	Playback Latency
HLS	Mature, well adapted to high-concurrency scenarios	SDK integration is required.	3s - 5s
FLV	Mature, well adapted to high-concurrency scenarios	SDK integration is required	2s - 3s
RTMP	Relatively low latency	Poor performance in high-concurrency scenarios	1s - 3s
WebRTC	Lowest latency	SDK integration is required	< 1s

Note:
LVB and LEB are priced differently. For details, please see LVB Billing Overview and LEB Billing Overview.

Notes

The Video Cloud SDK **does not impose any limit on the sources of playback URLs**, which means you can use it to play both Tencent Cloud and non-Tencent Cloud URLs. However, the player of the SDK supports only live streaming URLs in FLV, RTMP, HLS (M3U8), and WebRTC formats and VOD URLs in MP4, HLS (M3U8), and FLV formats.

Sample Code

Regarding frequently asked questions among developers, Tencent Cloud offers a straightforward API example project, which you can use to quickly learn how to use different APIs.

Platform	GitHub Address
iOS	Github
Android	Github
Flutter	Github

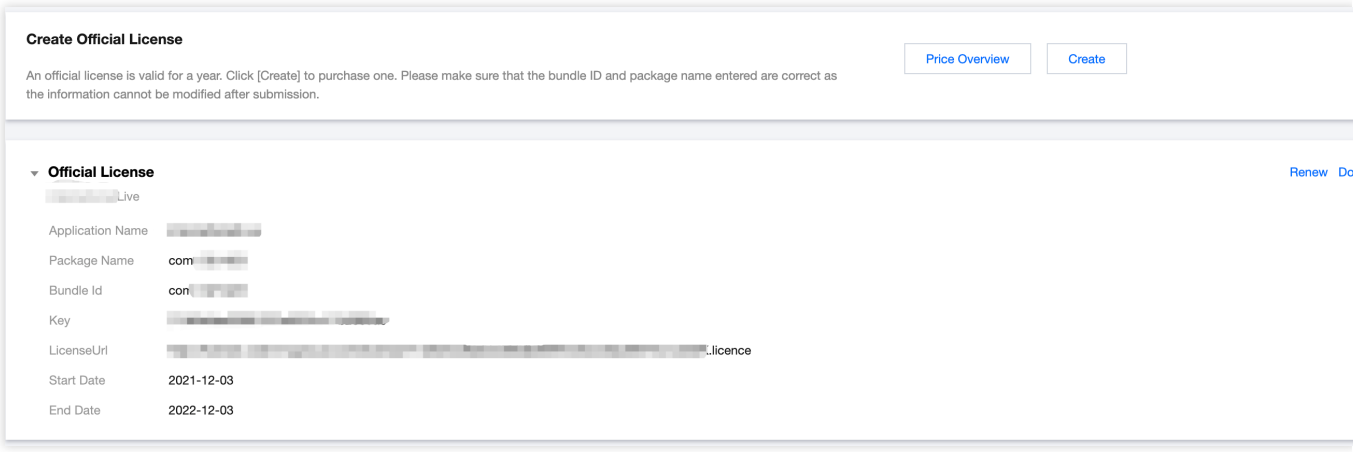
Integration

Step 1. Download the SDK

Download the SDK and follow the instructions in SDK Integration to integrate the SDK into your application.

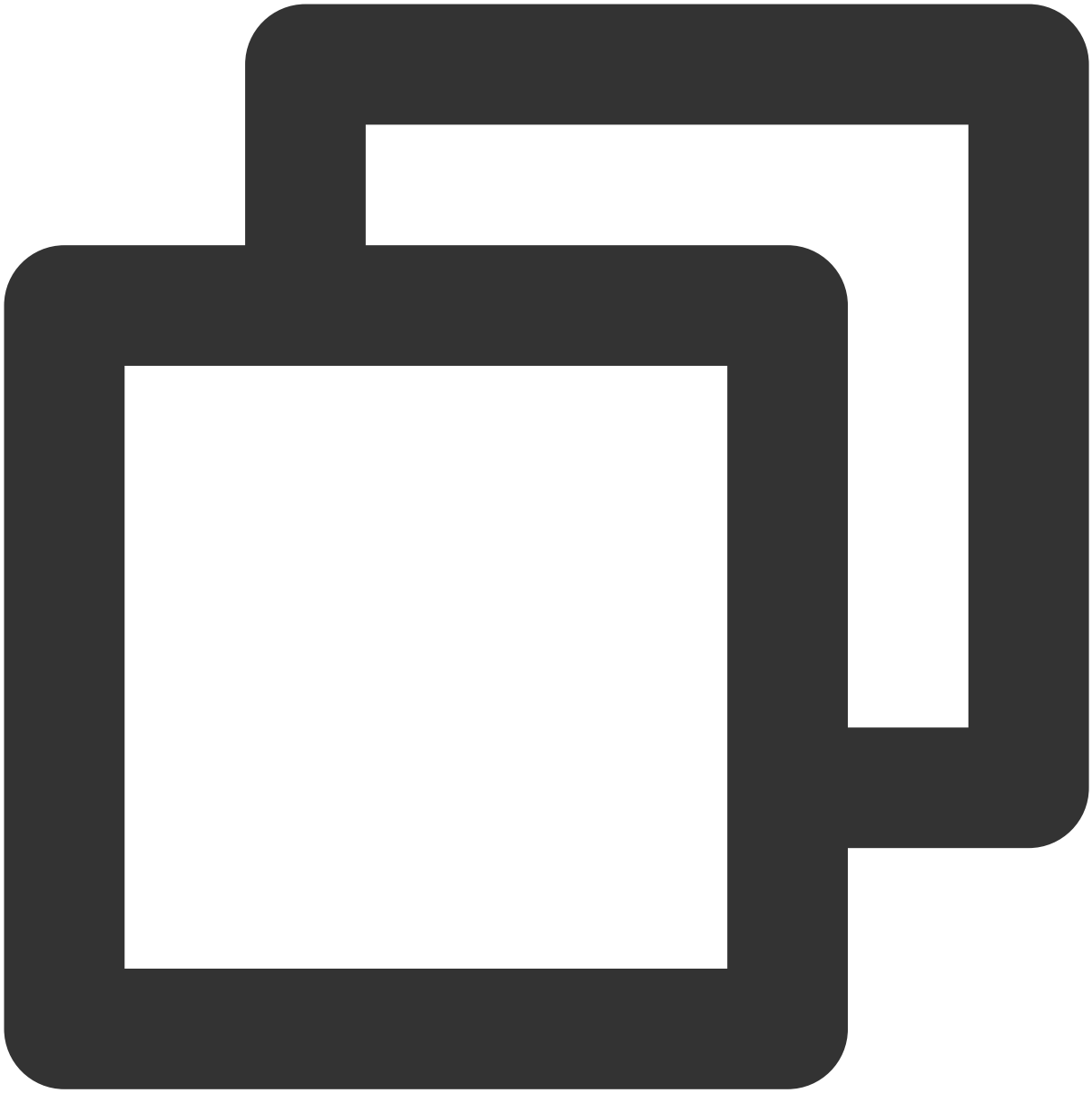
Step 2. Configure License Authorization for SDK

1. Obtain license authorization：
If you have obtained the relevant license authorization， need to Get License URL and License Key in Cloud Live Console



If you have not yet obtained the license authorization, Please reference Adding and Renewing Licenses to make an application.

2. Before your App calls SDK-related functions (it is recommended in the Application class), set the following settings:



```
public class MyApplication extends Application {

@Override
public void onCreate() {
super.onCreate();
String licenceURL = ""; // your licence url
String licenceKey = ""; // your licence key
V2TXLivePremier.setEnvironment("GDPR");  // set your environment
V2TXLivePremier.setLicence(this, licenceURL, licenceKey);
V2TXLivePremier.setObserver(new V2TXLivePremierObserver() {
        @Override
```

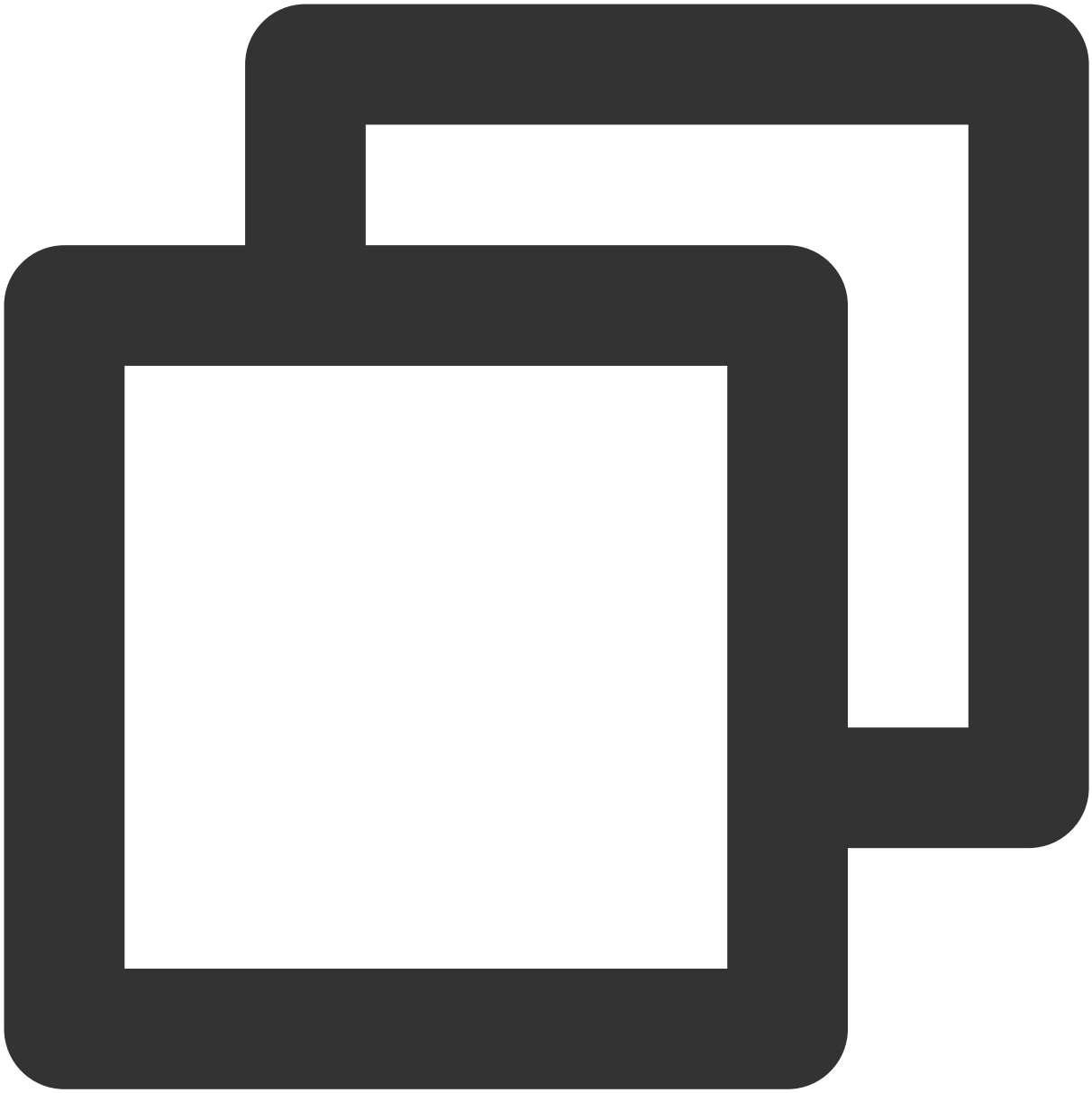
```
        public void onLicenceLoaded(int result, String reason) {
            Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reason);
        }
    });
}
```

Note:

The packageName configured in the license must be the same as the application itself, otherwise the play stream will fail.

Step 3. Create a rendering view

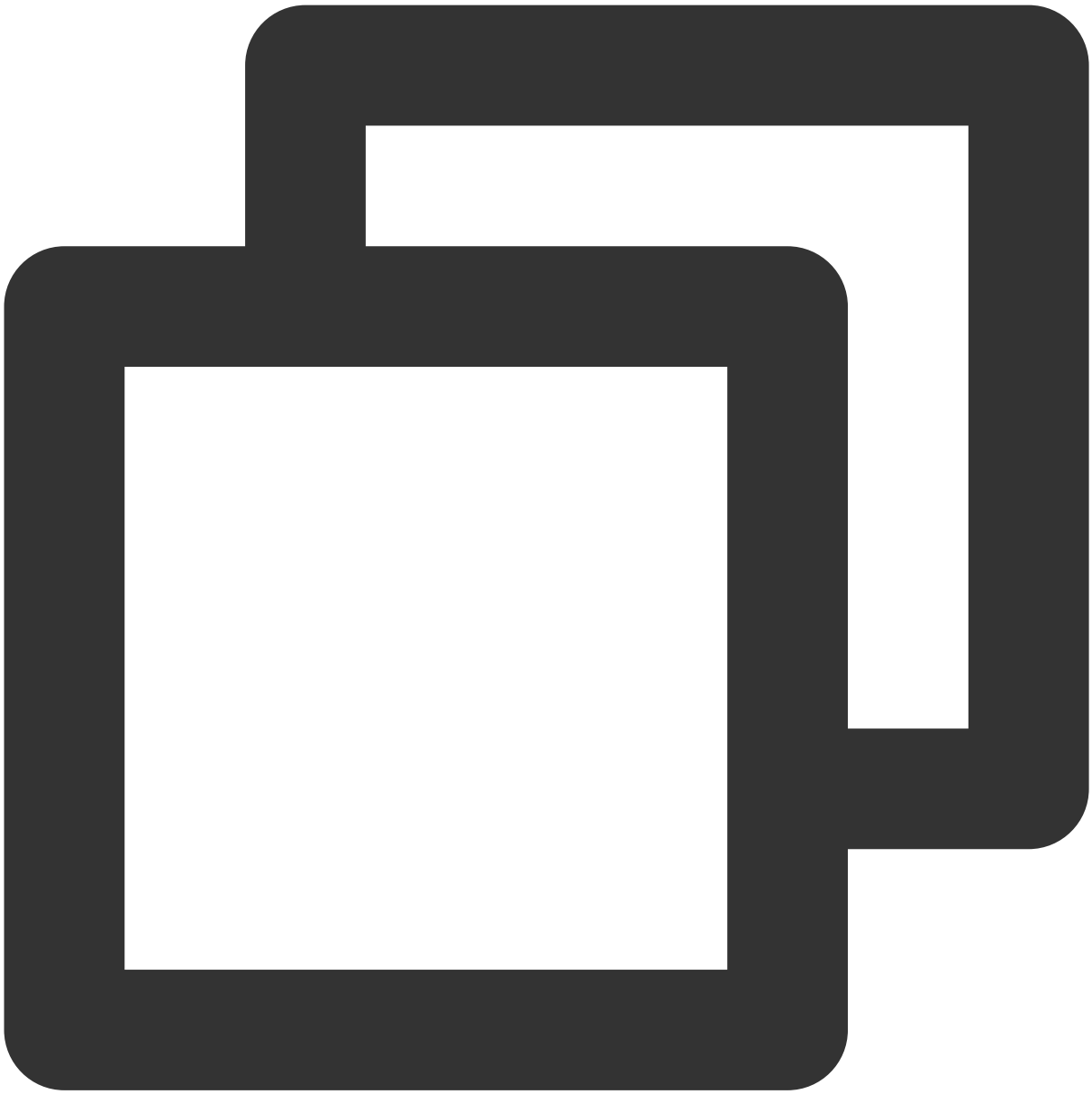
For the player to display video images, you need to add a rendering view in the layout XML file:



```
<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="visible"/>
```

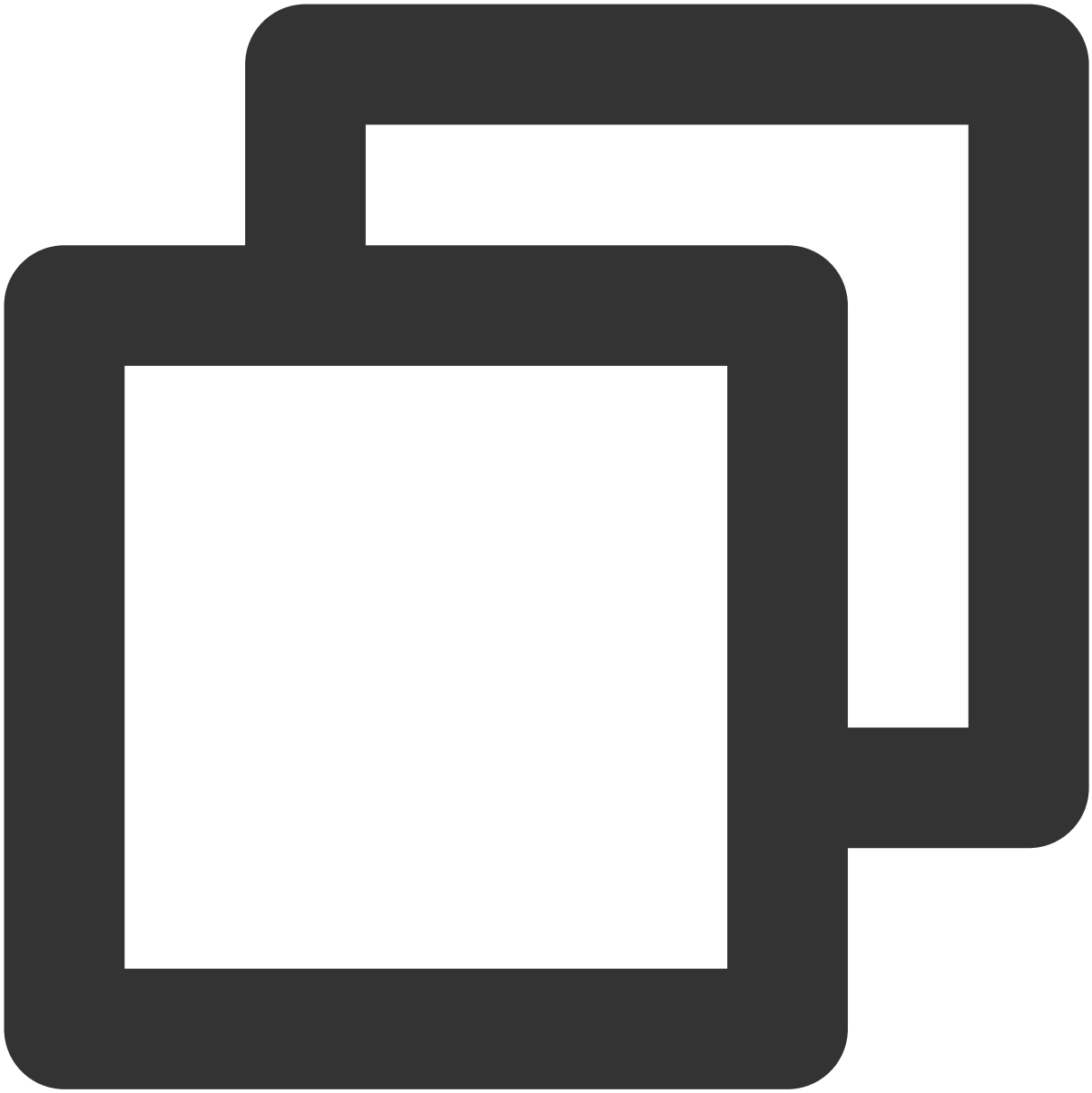
Step 4. Create a player object

The **V2TXLivePlayer** module in the Video Cloud SDK offers live playback capabilities. Use the **setRenderView** API to associate the module with the **video_view** control added to the UI in Step 3.



```
// mPlayerView is the view added in step 1
TXCloudVideoView mView = (TXCloudVideoView) view.findViewById(R.id.video_view);
// Create a player object
V2TXLivePlayer mLivePlayer = new V2TXLivePlayerImpl(mContext);
// Associate the player object with the view
mLivePlayer.setRenderView(mView);
```

Step 5. Start playback



```
String flvUrl = "http://2157.liveplay.myqcloud.com/live/2157_xxxx.flv";
mLivePlayer.startPlay(flvUrl);
```

Step 6. Change the fill mode

view: size and position

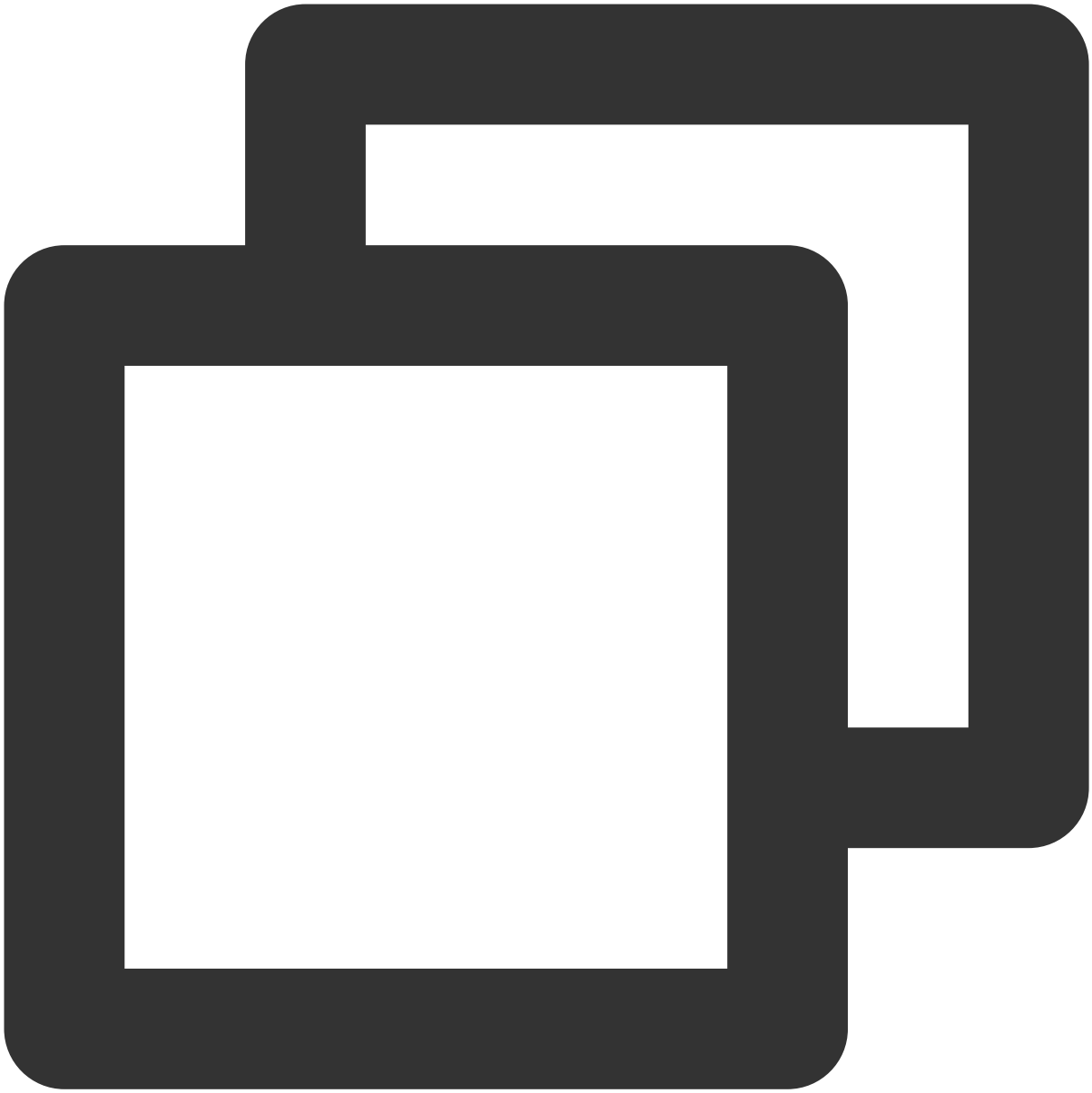
You can modify the size and position of video images by adjusting the size and position of the `video_view` control added in Step3.

setRenderFillMode: aspect fill or aspect fit

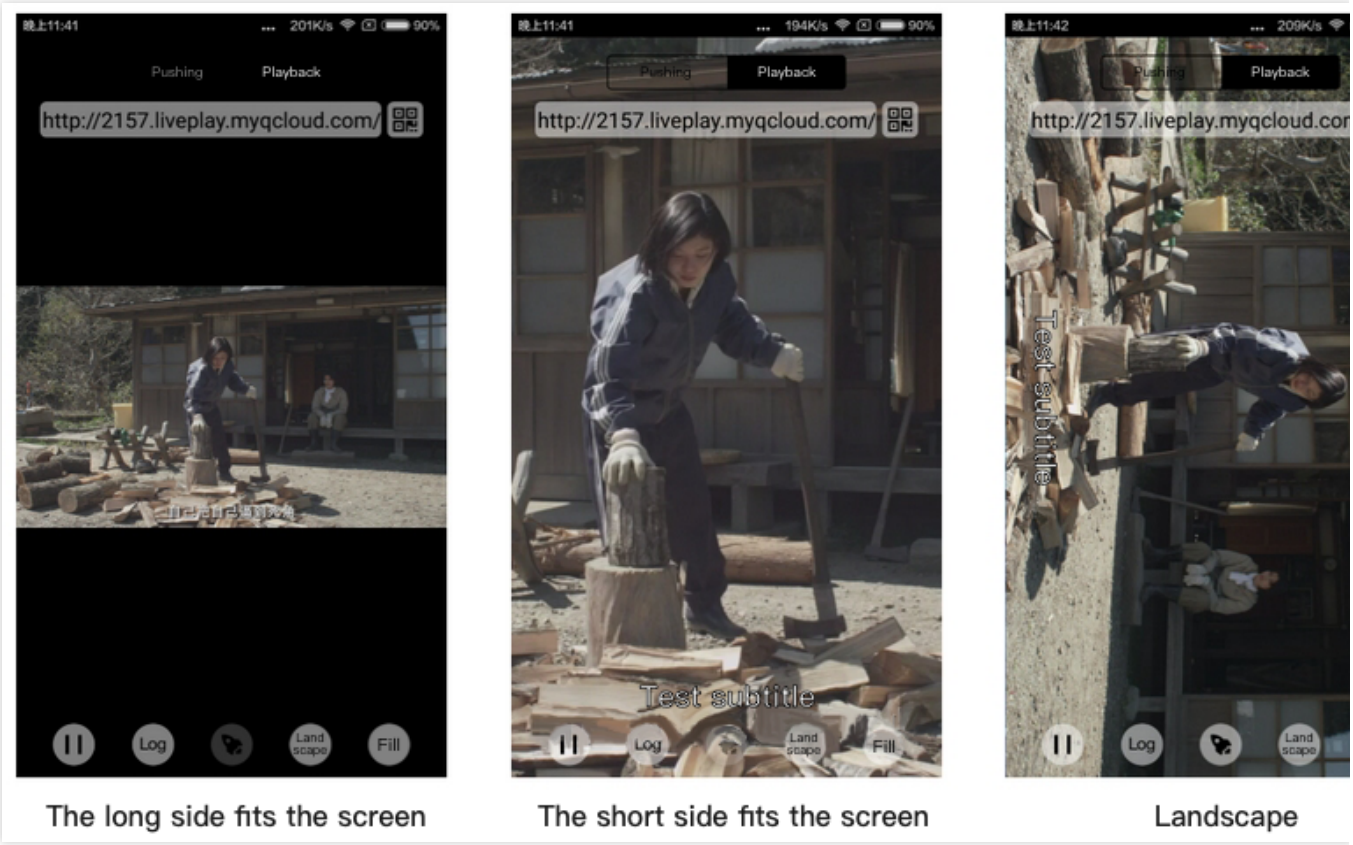
Value	Description
V2TXLiveFillModeFill	Images are scaled to fill the entire screen, and the excess parts are cropped. There are no black bars in this mode, but images may not be displayed in whole.
V2TXLiveFillModeFit	Images are scaled as large as the longer side can go. Neither side exceeds the screen after scaling. Images are centered, and there may be black bars.

setRenderRotation: clockwise rotation of video

Value	Description
V2TXLiveRotation0	Original
V2TXLiveRotation90	Rotate 90 degrees clockwise
V2TXLiveRotation180	Rotate 180 degrees clockwise
V2TXLiveRotation270	Rotate 270 degrees clockwise



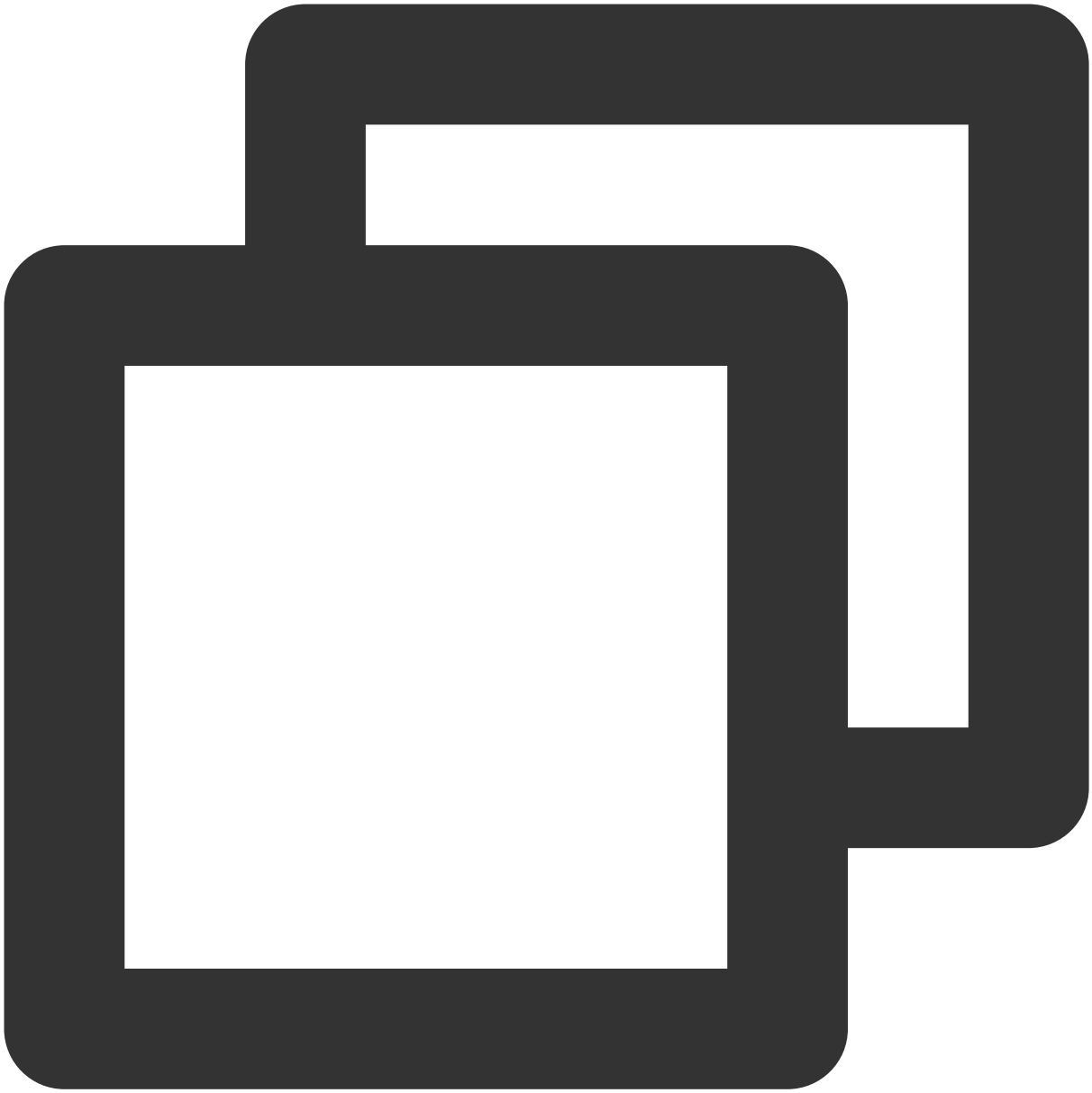
```
// Set the fill mode
mLivePlayer.setRenderFillMode(V2TXLiveFillModeFit);
// Set the rotation of video
mLivePlayer.setRenderRotation(V2TXLiveRotation0);
```



The long side fits the screen | The short side fits the screen | Landscape

Step 7. Pause playback

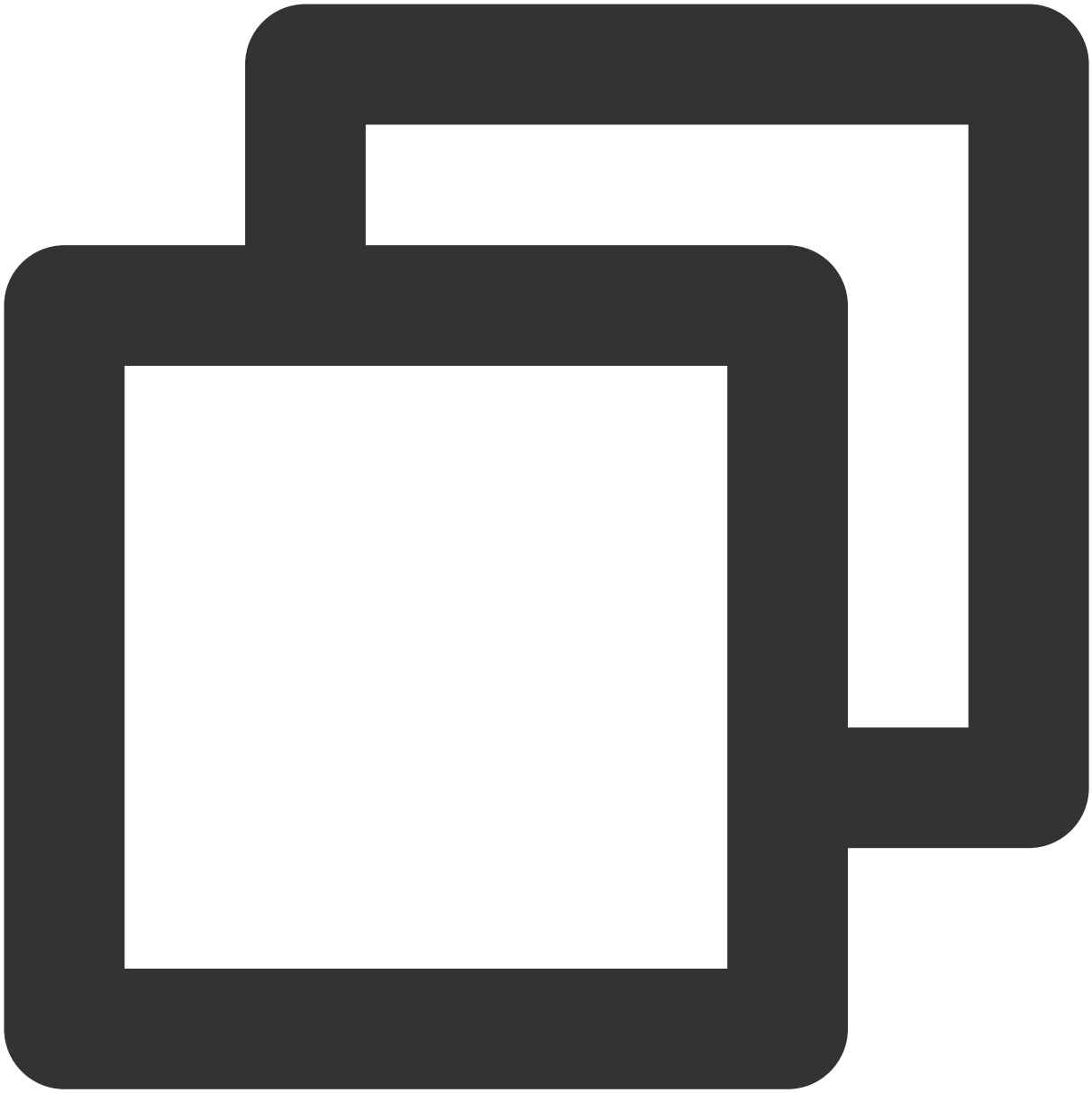
Technically speaking, you cannot pause a live playback. In this document, by pausing playback, we mean **freezing video** and **disabling audio**. In the meantime, new video streams continue to be sent to the cloud. When you resume playback, it starts from the time of resumption. This is in contrast to VOD. With VOD, when you pause and resume playback, the player behaves the same way as it does when you pause and resume a local video file.



```
// Pause playback
mLivePlayer.pauseAudio();
mLivePlayer.pauseVideo();
// Resume playback
mLivePlayer.resumeAudio();
mLivePlayer.resumeVideo();
```

Step 8. Stop playback

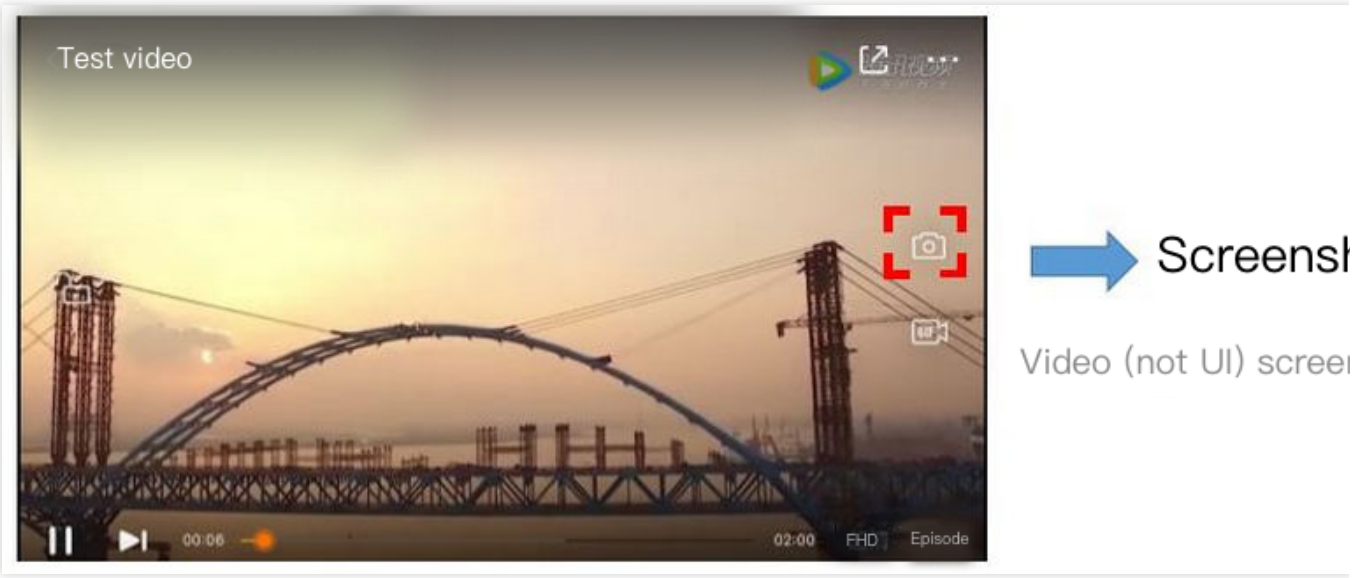
Call `stopPlay` to stop playback.

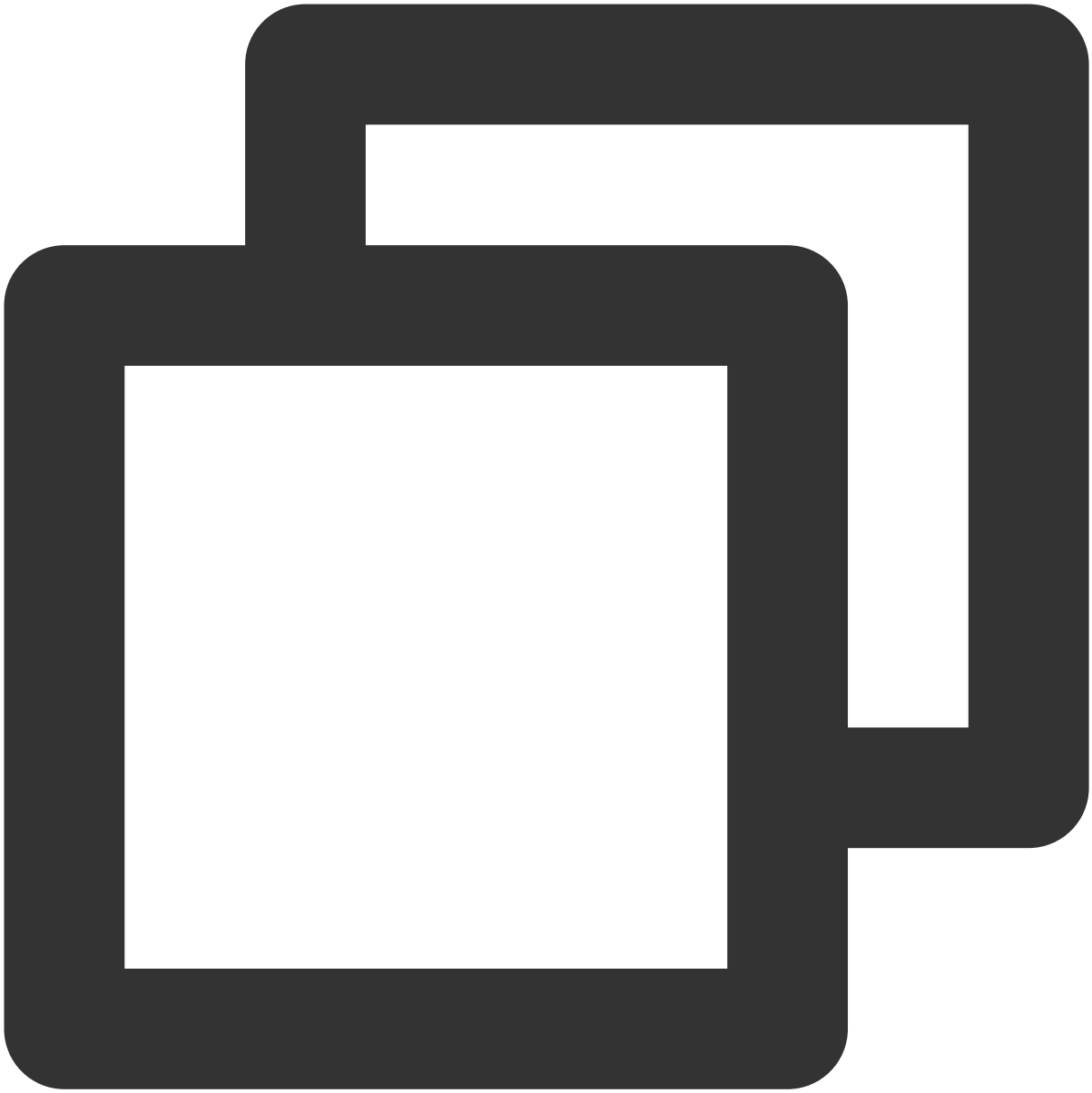


```
mLivePlayer.stopPlay();
```

Step 9. Take a screenshot

Call **snapshot** to take a screenshot of the live video streamed. This method captures a frame of the streamed video. To capture the UI, use the corresponding API of the Android system.





```
mLivePlayer.setObserver(new MyPlayerObserver());
mLivePlayer.snapshot();
// Get the screenshot taken in the onSnapshotComplete callback of MyPlayerObserver
private class MyPlayerObserver extends V2TXLivePlayerObserver  {
    ...
    @Override
    public void onSnapshotComplete(V2TXLivePlayer v2TXLivePlayer, Bitmap bitmap) {
    }
    ...
}
```

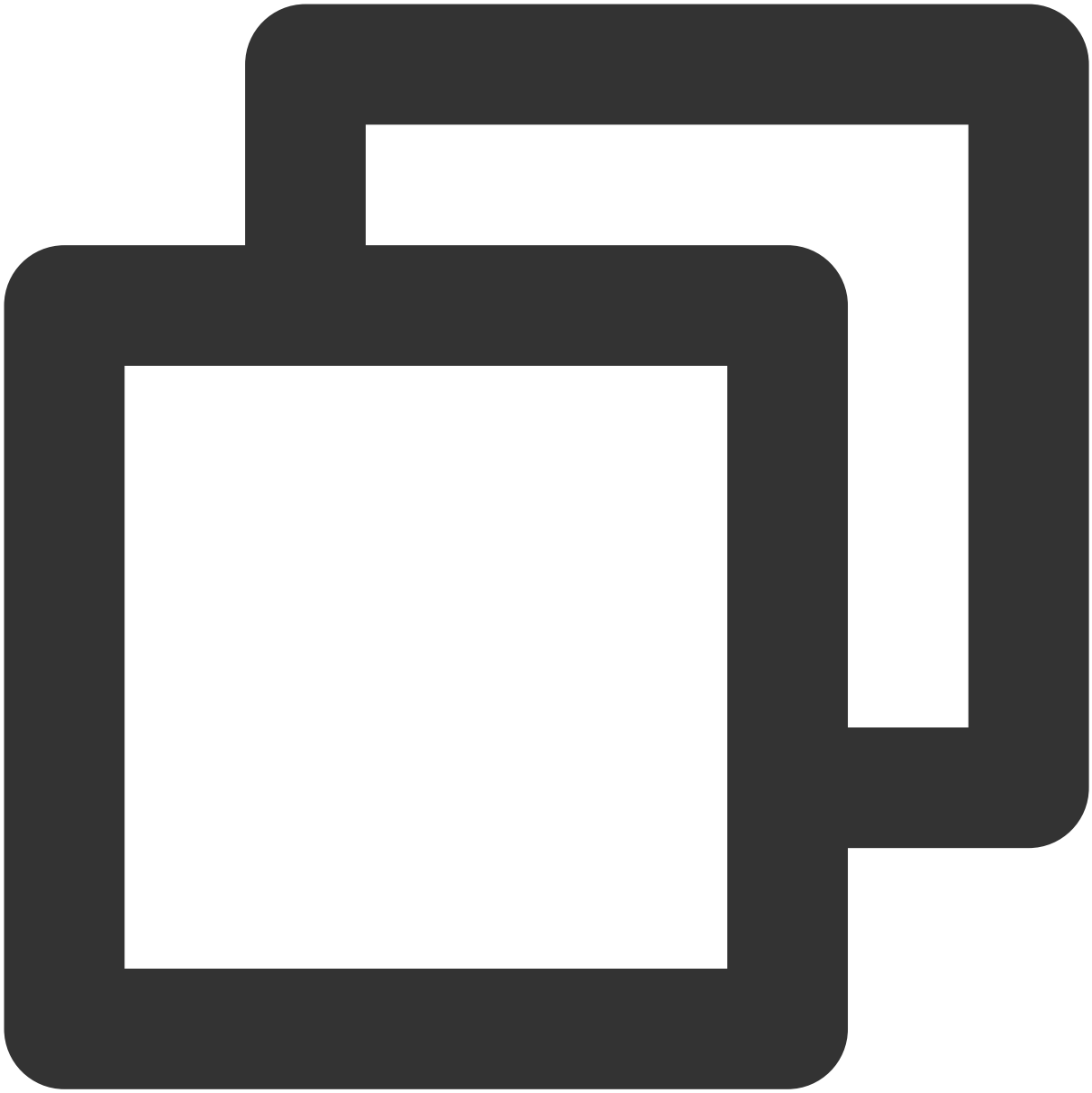
Latency Control

The live playback feature of the SDK is not based on FFmpeg, but Tencent Cloud's proprietary playback engine, which is why the SDK offers better latency control than open-source players do. We provide three latency control modes, which can be used for showrooms, game streaming, and hybrid scenarios.

Comparison of the three modes

Mode	Stutter	Average Latency	Scenario	Remarks
Speedy	More likely than the speedy mode	2-3s	Live showroom (Chongding Dahui)	The mode delivers low latency and is suitable for latency-sensitive scenarios.
Smooth	Least likely of the three	>= 5s	Game streaming (Penguin Esports)	Playback is least likely to stutter in this mode, which makes it suitable for ultra-high-bitrate streaming of games such as PUBG.
Auto	Self-adaptive to network conditions	2-8s	Hybrid	The better network conditions at the audience end, the lower the latency.

Code to integrate the three modes



```
// Auto mode
mLivePlayer.setCacheParams(1.0f, 5.0f);
// Speedy mode
mLivePlayer.setCacheParams(1.0f, 1.0f);
// Smooth mode
mLivePlayer.setCacheParams(5.0f, 5.0f);
// Start playback after configuration
```

Note:

For more information on stuttering and latency control, see Video Stutter.

Listening for SDK Events

You can bind a V2TXLivePlayerObserver to your `V2TXLivePlayer` object to receive callback notifications about the player status, playback volume, first audio/video frame, statistics, warning and error messages, etc.

Periodically triggered notification

The onStatisticsUpdate callback notification is triggered every 2 seconds to update you on the player's status in real time. Like a car's dashboard, the callback gives you information about the SDK, such as network conditions and video information.

Parameter	Description
appCpu	CPU usage (%) of the application
systemCpu	CPU usage (%) of the system
width	Video width
height	Video height
fps	Frame rate (fps)
audioBitrate	Audio bitrate (Kbps)
videoBitrate	Video bitrate (Kbps)

The onPlayoutVolumeUpdate callback, which notifies you of the player's volume, works only after you call enableVolumeEvaluation to enable the volume reminder. You can set the interval of the callback by specifying the `intervalMs` parameter of `enableVolumeEvaluation`.

Event-triggered notifications

Other callbacks are triggered when specific events occur.

LEB

Last updated：2024-01-13 15:49:41

LEB Overview

Live Event Broadcasting (LEB) is the ultra-low-latency version of LVB. It features lower latency than traditional streaming protocols and delivers superior playback experience with millisecond latency. It is suitable for scenarios with high requirements on latency, such as online education, sports streaming, and online quizzes.

Note:

The figure above (made using scrcpy) is a comparison of LEB and standard CDN live streaming. The images on the left and in the middle show the playback end of standard CDN live streaming and **LEB**, and the image on the right shows the publishing end.

LVB and LEB are priced differently. For details, please see LVB Billing Overview and LEB Billing Overview.

Strengths

Strength	Description
Playback with millisecond latency	The latency is kept within 1s thanks to the use of UDP, as opposed to 3-5s in traditional live streaming. This, along with excellent instant streaming performance and low stuttering rate, guarantees a superior streaming experience.
Diverse features and smooth migration	LEB integrates a wide range of features including stream publishing, transcoding, recording, screenshot, porn detection, and playback. It allows smooth migration from standard live streaming.
Easy-to-use, secure, and reliable	The use of a standard protocol makes integration easy. You can play live video on Chrome and Safari without installing any plugins. In addition, the playback protocol encrypts video by default for improved security and reliability.

Use cases

Scenario	Description
Sports event	LEB offers ultra-low-latency streaming for sports events. It brings sports content to audience at low latency, allowing audience to learn what's happening in real time.
E-commerce streaming	Some e-commerce streaming scenarios, for example, online auctions and

	sales promotion, require extremely low latency. LEB's ability to stream at ultra-low latency ensures that hosts and audience get real-time feedback from each other, improving online shopping experience.
Online classes	LEB can be used for online classes. Its ability to stream at ultra-low latency allows teachers and students to interact with each other as they do in offline classes.
Online quizzes	Due to latency, some online quizzes have to insert extra frames at the audience end to ensure that the host and audience are in sync with each other. This is not necessary if you use LEB, whose ultra-low-latency streaming capability makes sure that the two sides are in sync. It helps you implement online quizzes more easily and deliver smoother experience.
Showrooms	LEB can significantly improve the experience of latency-sensitive interactions such as gift giving in live showrooms.

Tryout

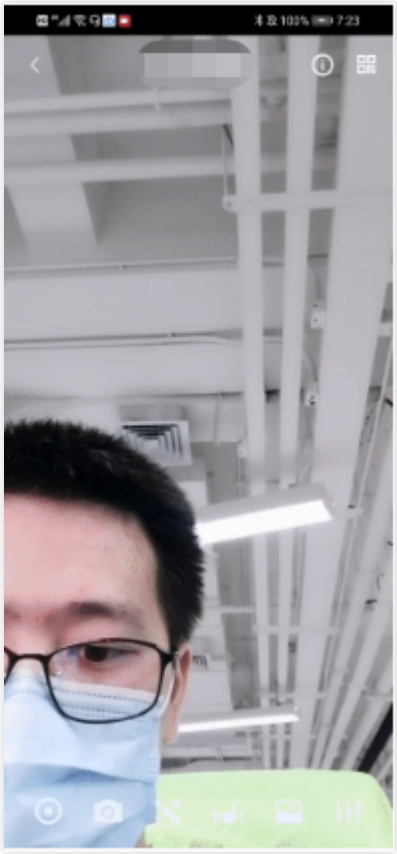
Video Cloud Toolkit is a comprehensive audio-video solution developed by Tencent Cloud that allows you to try out the features of the TRTC, MLVB and UGC SDKs, including the **LEB Player**.

Note:

The demonstration and directions in this document use the demo app for Android as an example. The UI of the app for iOS is slightly different.

Source code and demonstration

Source Code	Demo	Publishing Demonstration (Android)	F
Android			

iOS		

Note:

In addition to the above sample code, regarding frequently asked questions among developers, Tencent Cloud offers a straightforward API example project, which you can use to quickly learn how to use different APIs.

iOS：MLVB-API-Example

Android：MLVB-API-Example

Flutter：Live-API-Example

Publishing

LEB is compatible with LVB, which means you can publish streams using an ordinary publisher and play the streams using LEB.

1. Download and install Video Cloud Toolkit, log in, and go to **MLVB > Push (Camera)**.
2. Allow the permissions asked, and tap **Auto-generate** to start publishing streams.
3. If publishing is successful, tap the QR code icon in the top right and select **LEB** to get the playback URL for LEB.
4. During publishing, you can tap the menu icon in the bottom right to apply filters, add background music, switch cameras, etc.

Playback

1. Download and install Video Cloud Toolkit, log in, and go to **Live Broadcast > LEB Player**.
2. Allow the permissions asked, tap the scan button, and scan the LEB playback URL obtained earlier.
3. Playback starts automatically once the QR code is read. During playback, you can tap the menu icon in the bottom right to mute playback or change other settings.

Integration

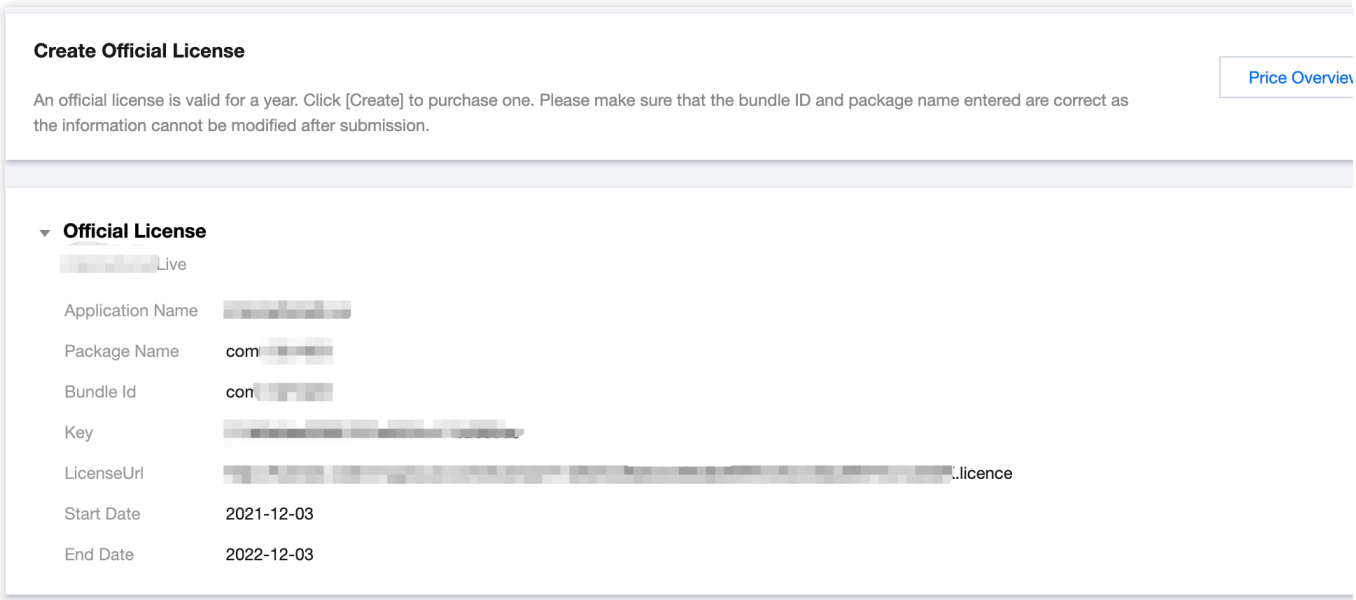
In the new version of the MLVB SDK, you can use V2TXLivePlayer for LEB and `V2TXLivePusher` for publishing. LEB supports WebRTC protocols and uses the standard extension method. All URLs in LEB start with `webrtc://` .

Step 1. Download the SDK

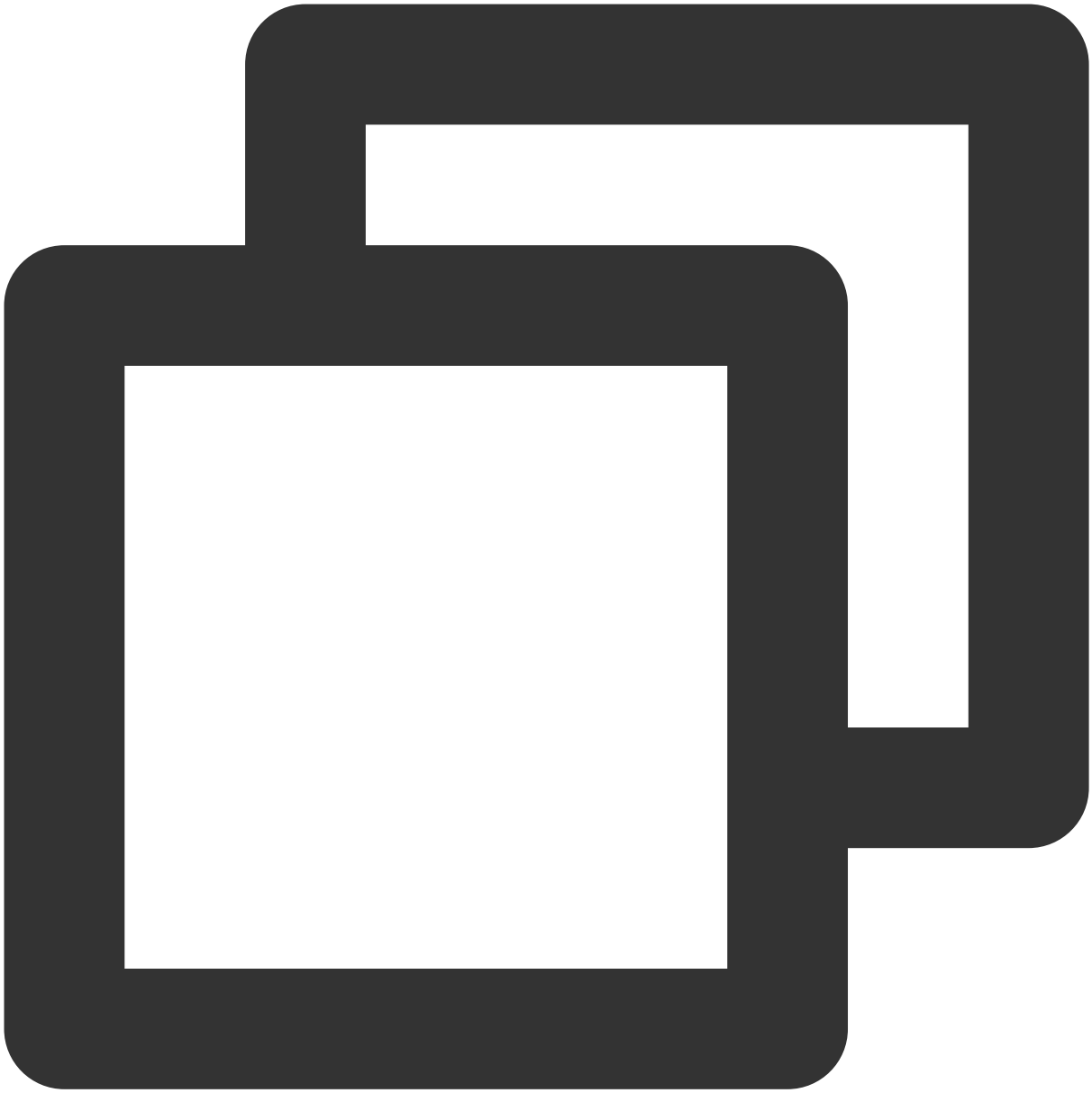
Download LiteAV_All or Professional at SDK Download.

Step 2. Configure License Authorization for SDK

1. Obtain license authorization：
If you have obtained the relevant license authorization， need to Get License URL and License Key in Cloud Live Console



If you have not yet obtained the license authorization， Please reference Adding and Renewing Licenses to make an application.
2. Before your App calls SDK-related functions (it is recommended in the Application class), set the following settings:



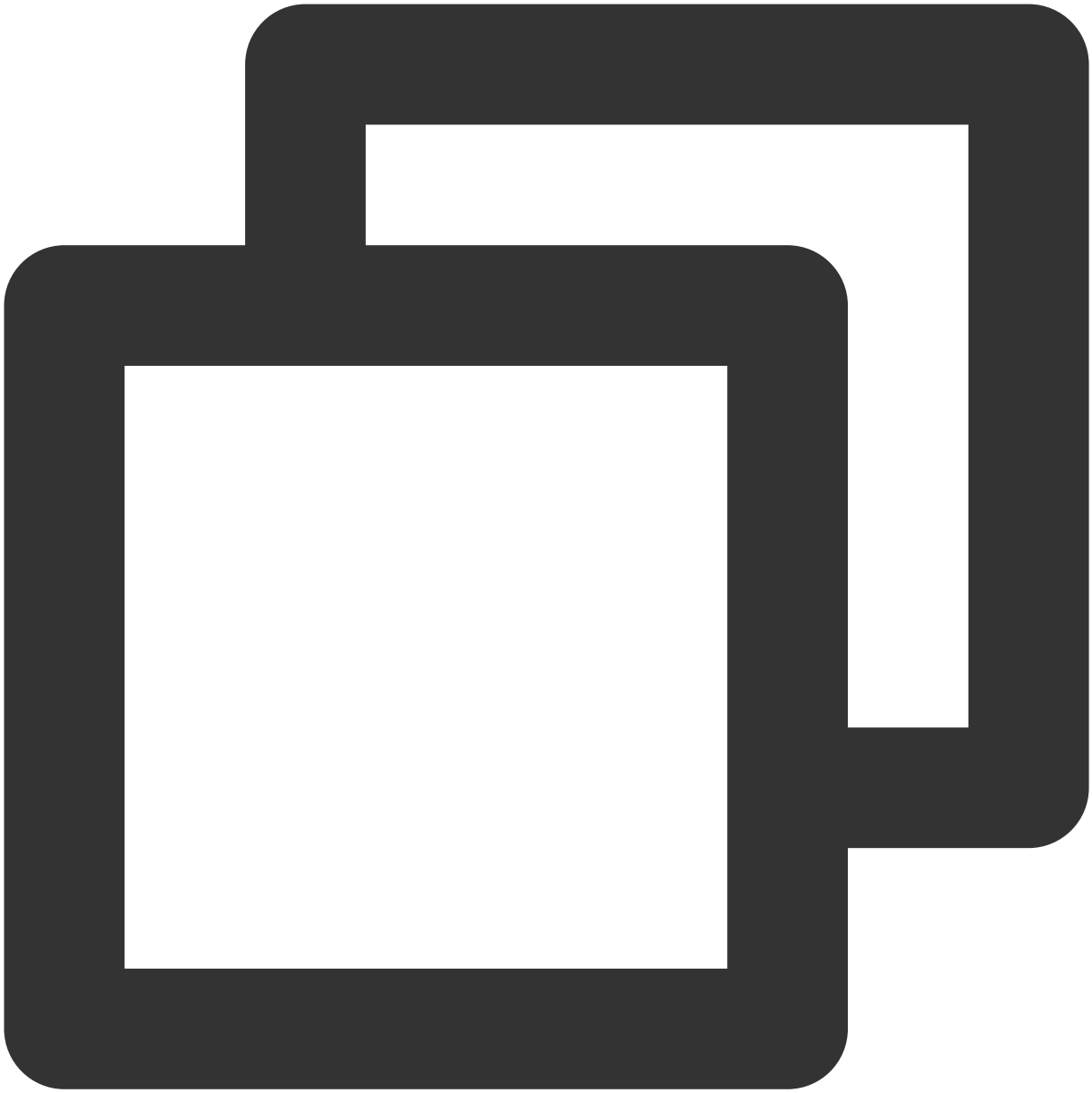
```
public class MApplication extends Application {

@Override
public void onCreate() {
    super.onCreate();
    String licenceURL = ""; // your licence url
    String licenceKey = ""; // your licence key
    V2TXLivePremier.setEnvironment("GDPR");  // set your environment
    V2TXLivePremier.setLicence(this, licenceURL, licenceKey);
    V2TXLivePremier.setObserver(new V2TXLivePremierObserver() {
            @Override
```

```
            public void onLicenceLoaded(int result, String reason) {
                Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reas
            }
        });
}
```

Step 3. Get a playback URL

In live streaming, URLs are needed for both publishing and playback. For instructions on how to get URLs for LEB, please see Live Event Broadcasting (LEB) > Get Playback URL.

All LEB URLs start with `webrtc://` , as in:



```
webrtc://{Domain}/{AppName}/{StreamName}
```

The table below lists the key fields in an LEB URL and their meanings.

Field	Description
webrtc://	Prefix
Domain	Domain name
AppName	Application name, which is `live` by default. It specifies the storage path of a live streaming

	file.
StreamName	Stream name, which is the unique identifier of a stream

Note:

To publish streams, please see Publishing from Camera or Publishing from Screen.

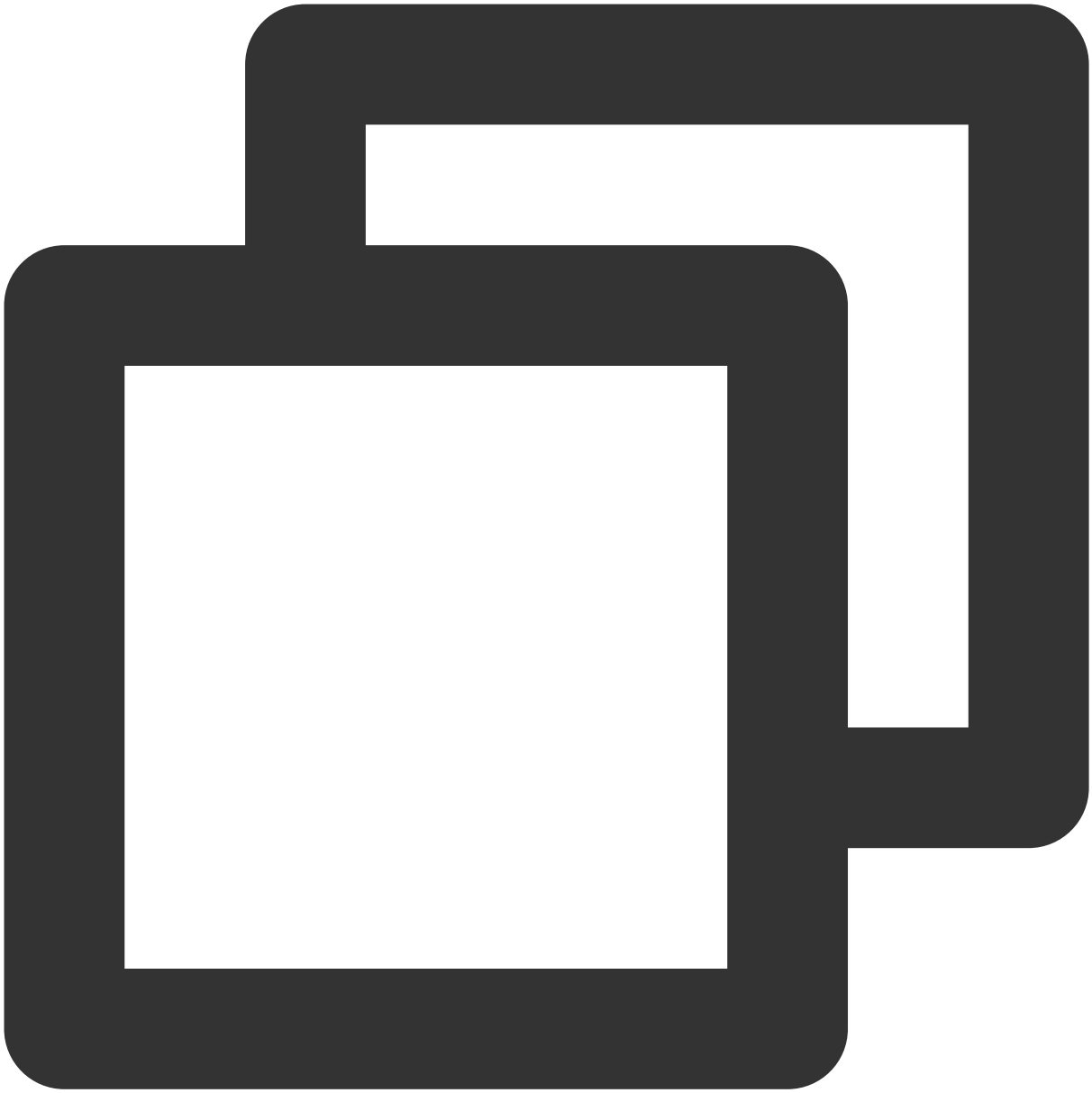
Step 4. Start LEB

You can use a `V2TXLivePlayer` object for LEB. For details, see the code below (make sure that you pass in the correct URL).

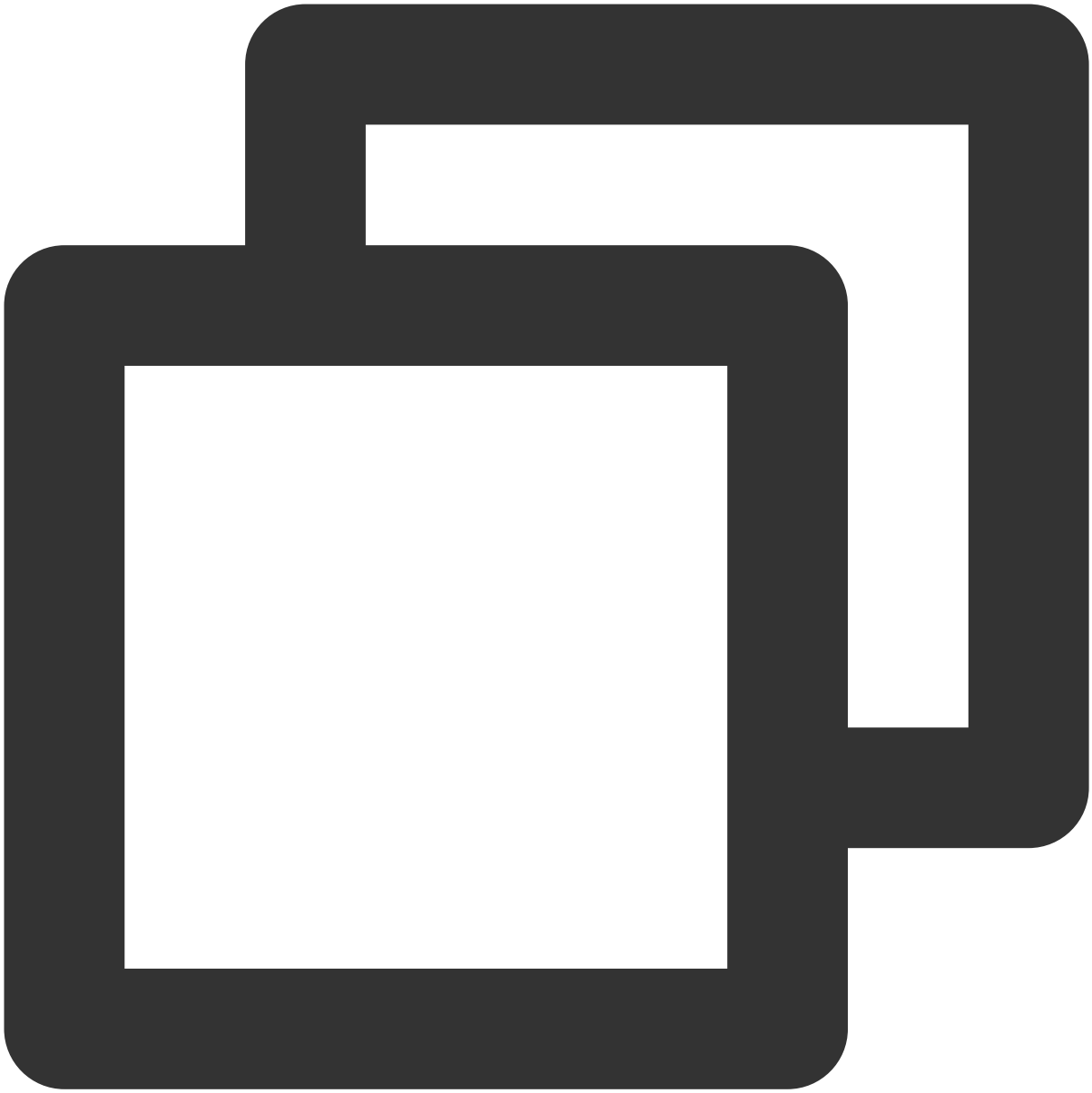
Sample code

Android

iOS



```
// Create a V2TXLivePlayer object
V2TXLivePlayer player = new V2TXLivePlayerImpl(mContext);
player.setObserver(new MyPlayerObserver(playerView));
player.setRenderView(mSurfaceView);
// Pass in the low-latency playback URL to start playback
player.startPlay("webrtc://{Domain}/{AppName}/{StreamName}");
```



```
V2TXLivePlayer *player = [[V2TXLivePlayer alloc] init];
[player setObserver:self];
[player setRenderView:videoView];
[player startPlay:@"webrtc://{Domain}/{AppName}/{StreamName}"];
```

Flutter
LVB

Last updated：2024-01-13 15:49:41

Basics

This document describes the live playback feature of Live Flutter Plugin.

Live streaming overview

In **live streaming**, the video source is pushed by the host in real time. When the host stops pushing the source, 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.

Supported protocols

Common live streaming protocols are as listed below. We recommend you use an FLV-based live streaming URL that starts with `http` and ends with `.flv` for LVB. For LEB, we recommend you use the WebRTC protocol. For more information, see LEB.

Protocol	Advantage	Disadvantage	Playback Latency
HLS	Mature, well adapted to high-concurrency scenarios	SDK integration is required	3–5 seconds
FLV	Mature, well adapted to high-concurrency scenarios	SDK integration is required.	2-3 seconds
RTMP	Relatively low latency	Poor performance in high-concurrency scenarios	1-3 seconds
WebRTC	Lowest latency	SDK integration is required	< 1 second

Note:
LVB and LEB are priced differently. For details, see LVB Billing Overview and LEB Billing Overview.

Notes

The SDK **does not impose any restrictions on the sources of playback URLs**, which means you can use it to play both Tencent Cloud and non-Tencent Cloud URLs. However, the player of the SDK supports only live streaming URLs in FLV, RTMP, HLS (M3U8), and WebRTC formats and VOD URLs in MP4, HLS (M3U8), and FLV formats.

Sample Code

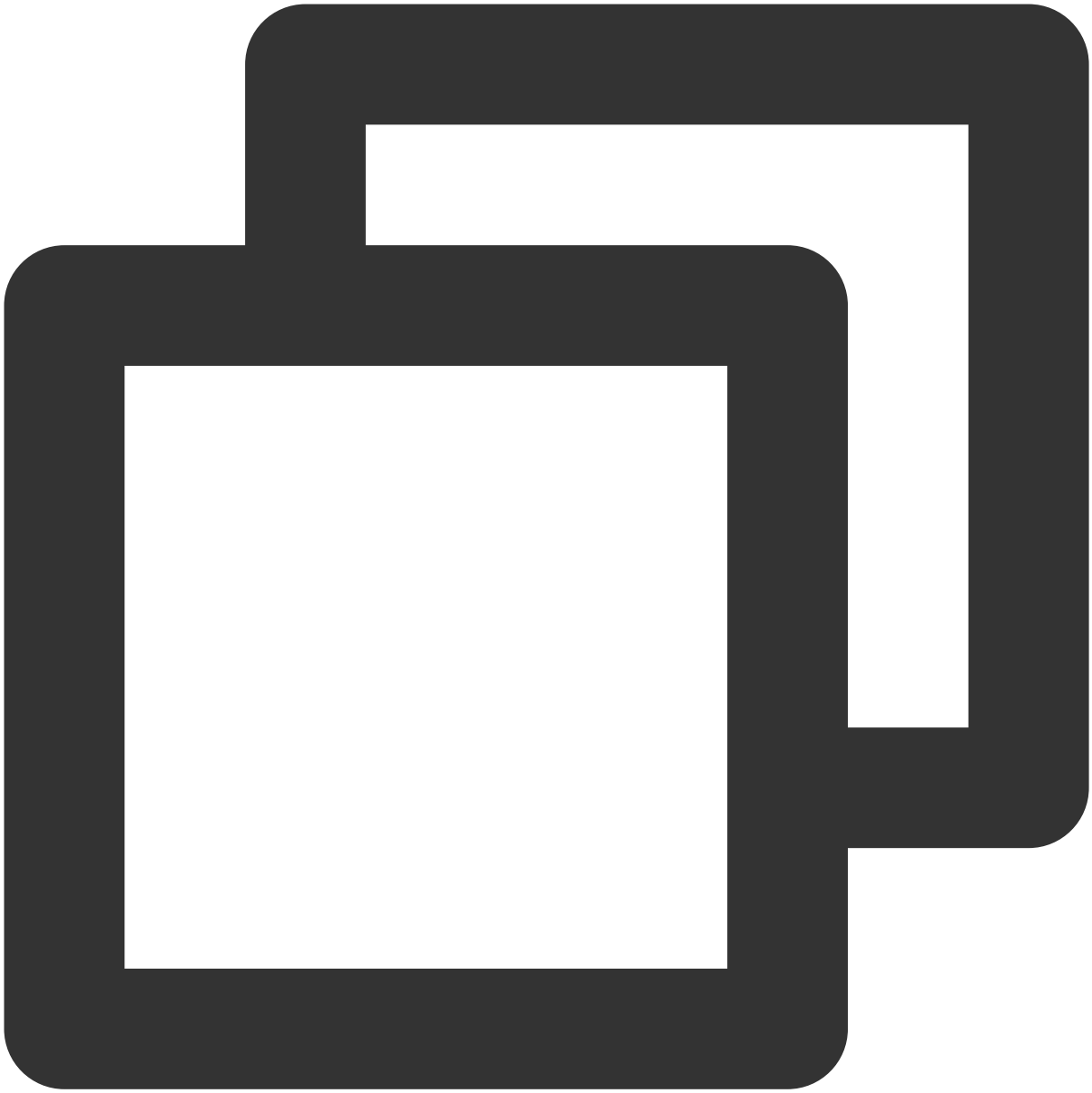
Tencent Cloud offers an easy-to-understand API example project to help you quickly learn how to use different APIs.

Platform	GitHub Address
iOS	GitHub
Android	GitHub
Flutter	live_flutter_plugin

Getting Started

1. Set dependencies

Integrate `live_flutter_plugin` into your application as instructed in SDK Integration Guide.

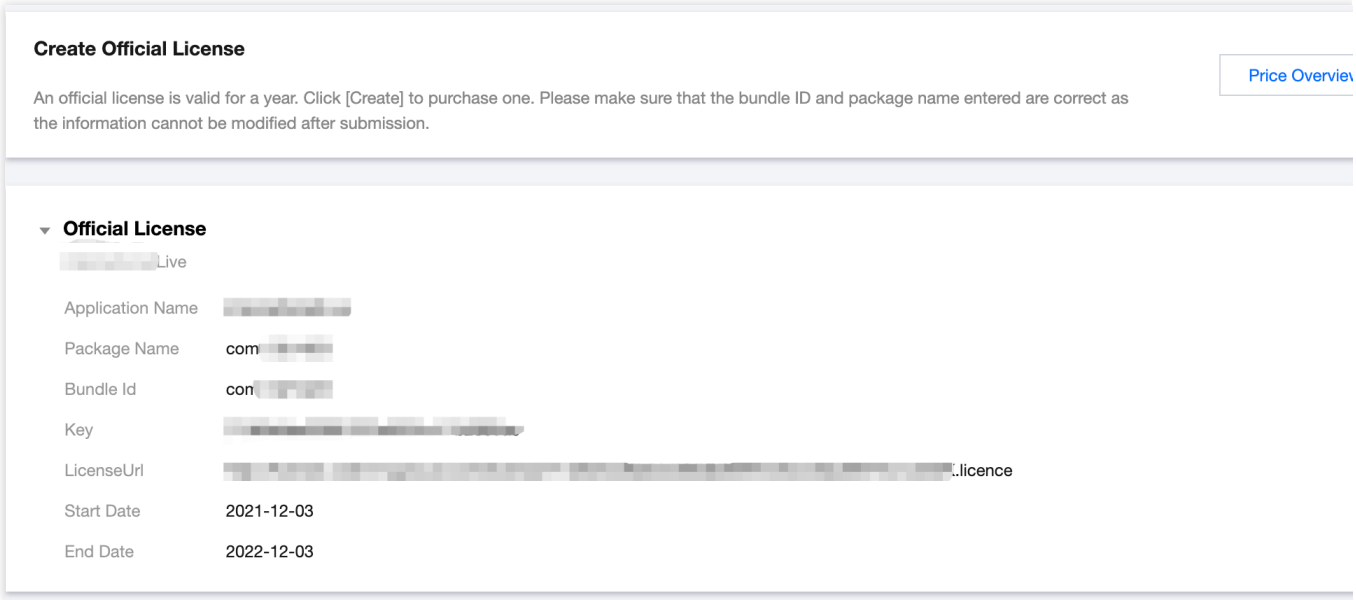


```
dependencies:
  live_flutter_plugin: latest version number
```

2. Configure a license for the SDK

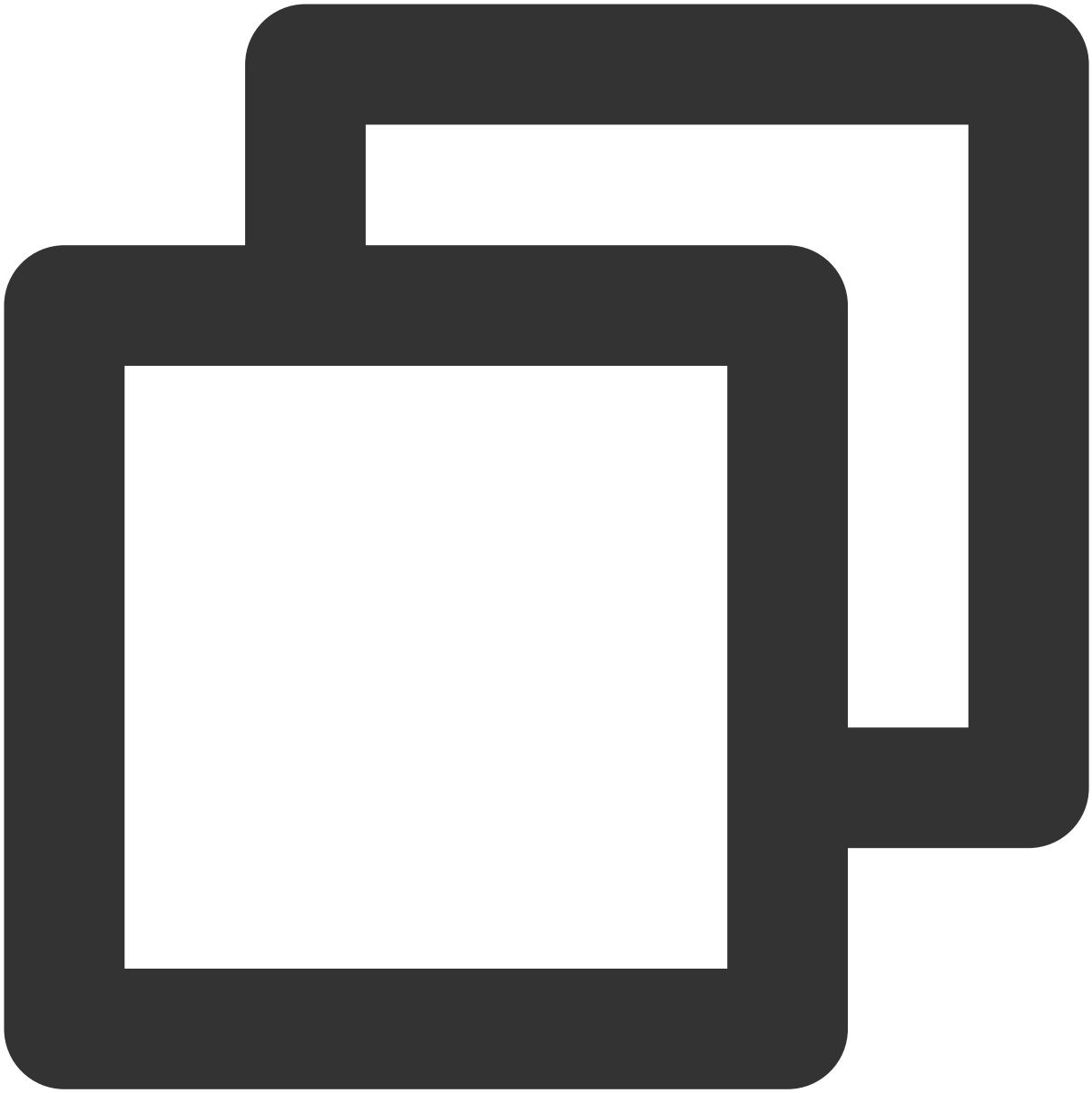
1. Get the license:

If you have the required license, get the license URL and key in the CSS console.



If you don't have the required license, apply for a license as instructed in New License and Renewal.

2. Before your application calls features of `live_flutter_plugin`, complete the following configuration:



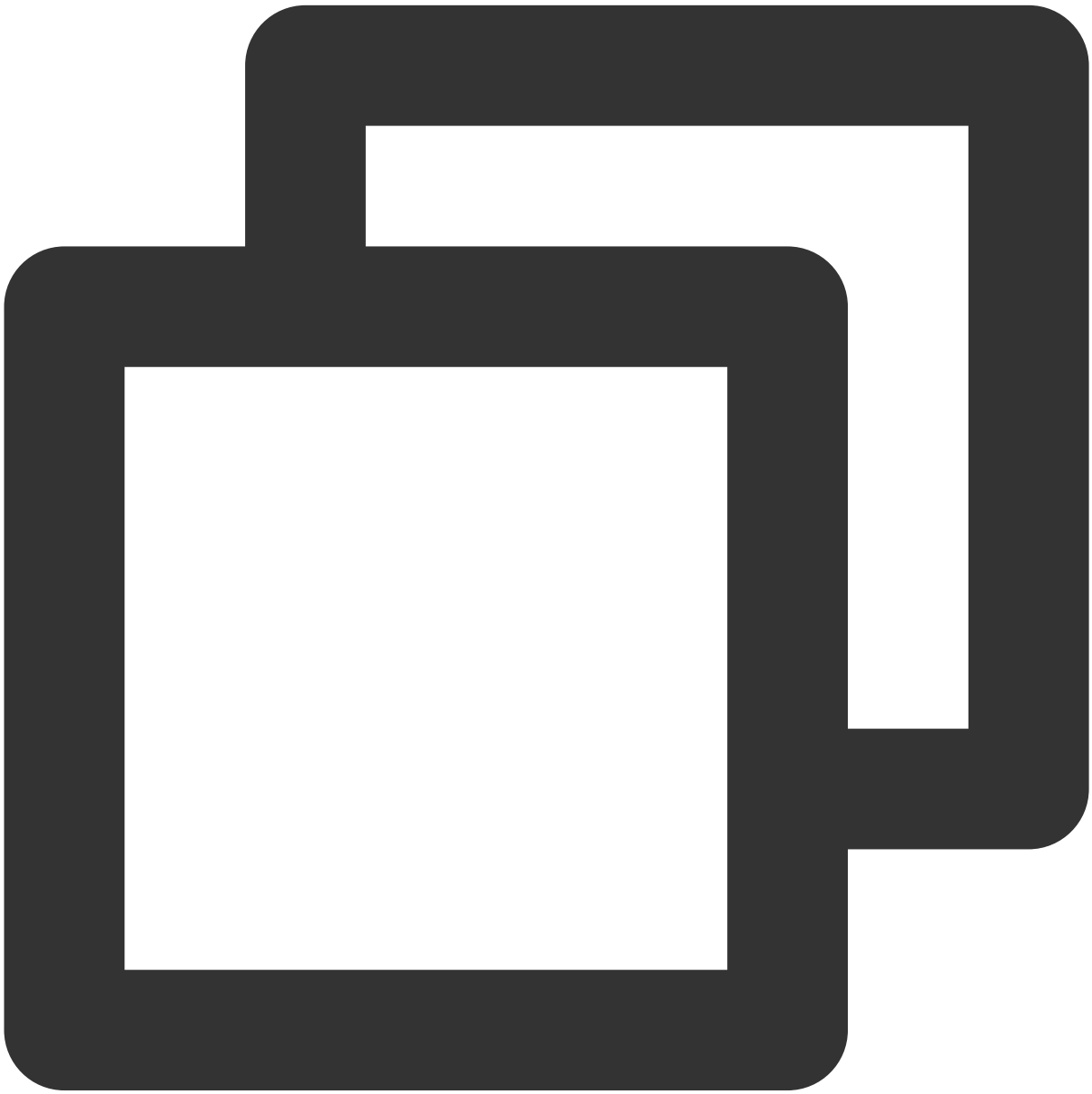
```
import 'package:live_flutter_plugin/v2_tx_live_premier.dart';

 /// Tencent Cloud license management page (https://console.intl.cloud.tencent.com/
setupLicense() {
  // The license URL of the current application
  var LICENSEURL = "";
  // The license key of the current application
  var LICENSEURLKEY = "";
  V2TXLivePremier.setLicence(LICENSEURL, LICENSEURLKEY);
}
```

Note:

The `packageName/BundleId` configured in the license must be the same as that of the application; otherwise, playback will fail.

3. Create the player

The `V2TXLivePlayer` module in the SDK offers live playback capabilities.

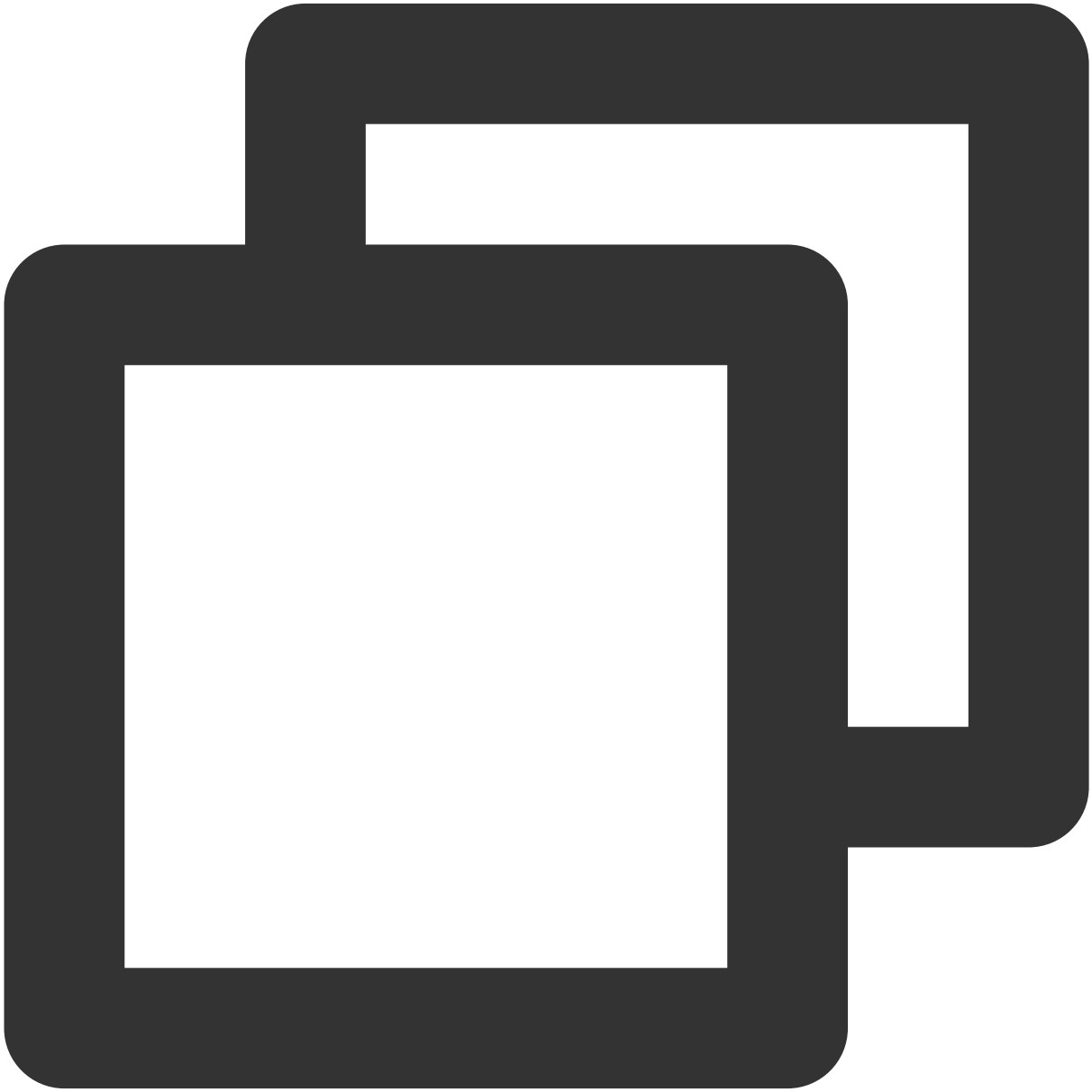


```
import 'package:live_flutter_plugin/v2_tx_live_player.dart';
/// Initialize `V2TXLivePlayer`
initPlayer() {
```

```
    _livePlayer = V2TXLivePlayer();
    _livePlayer.addListener(onPlayerObserver);
  }
```

4. Render the view

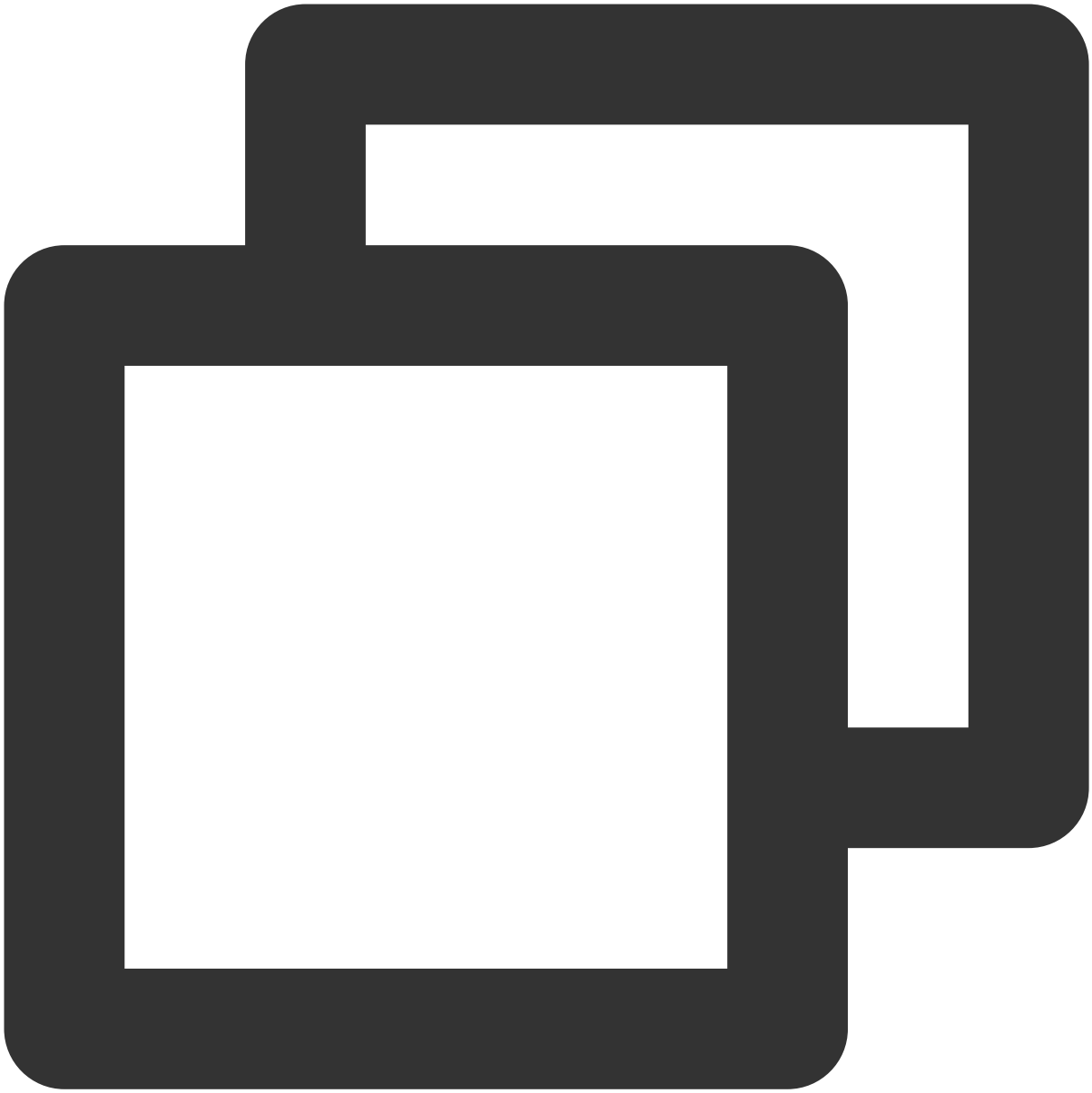
In Flutter, you need to depend on `v2_tx_live_video_widget` to create a video rendering view for the player to display video images on.



```
import 'package:live_flutter_plugin/widget/v2_tx_live_video_widget.dart';
```

```
/// The video rendering view widget
Widget renderView() {
  return V2TXLiveVideoWidget(
    onViewCreated: (viewId) async {
      // Set the video rendering view
      _livePlayer.setRenderViewID(_renderViewId);
    },
  );
}
```

5. Start playback

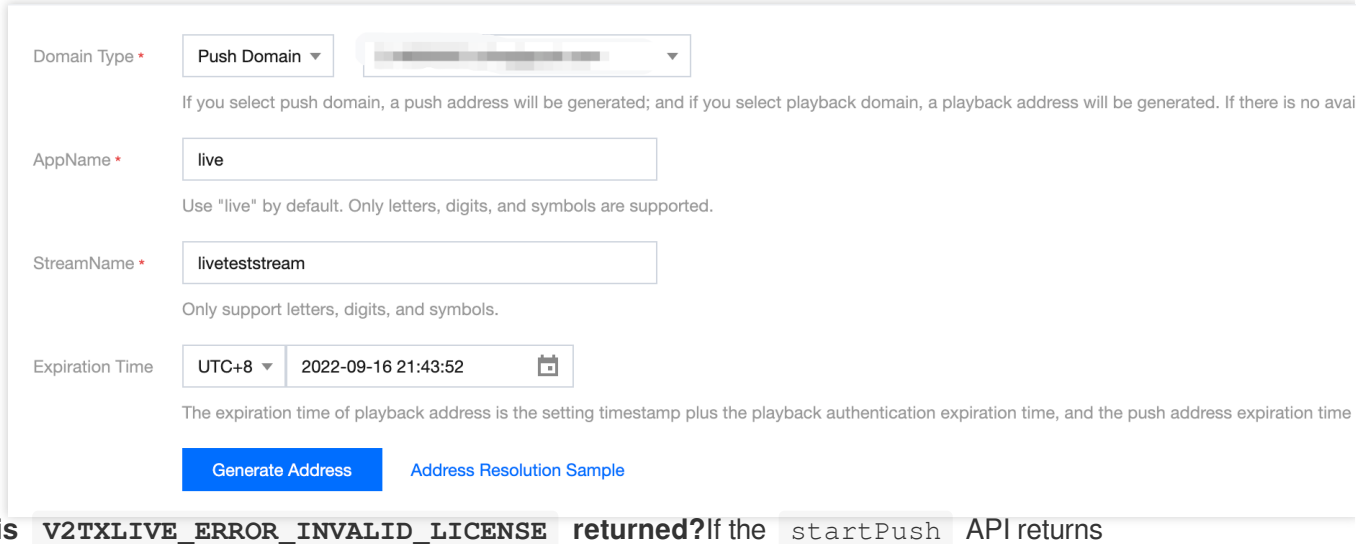


```
/// Start pulling the stream
startPlay() async {
  // Generate the `url` (RTMP/TRTC/LEB)
  var url = ""
  // Start pulling the stream
  await _livePlayer?.startPlay(url);
}
```

How do I get a valid stream push URL?

Activate CSS. In the **CSS console**, go to **Auxiliary Tools** > **Address Generator** to generate a stream push URL.

For more information, see Publishing/Playback URL.



Why is `V2TXLIVE_ERROR_INVALID_LICENSE` returned? If the `startPush` API returns `V2TXLIVE_ERROR_INVALID_LICENSE`, it means your license verification failed. Please check your configuration against Step 2. Configure a license for the SDK.

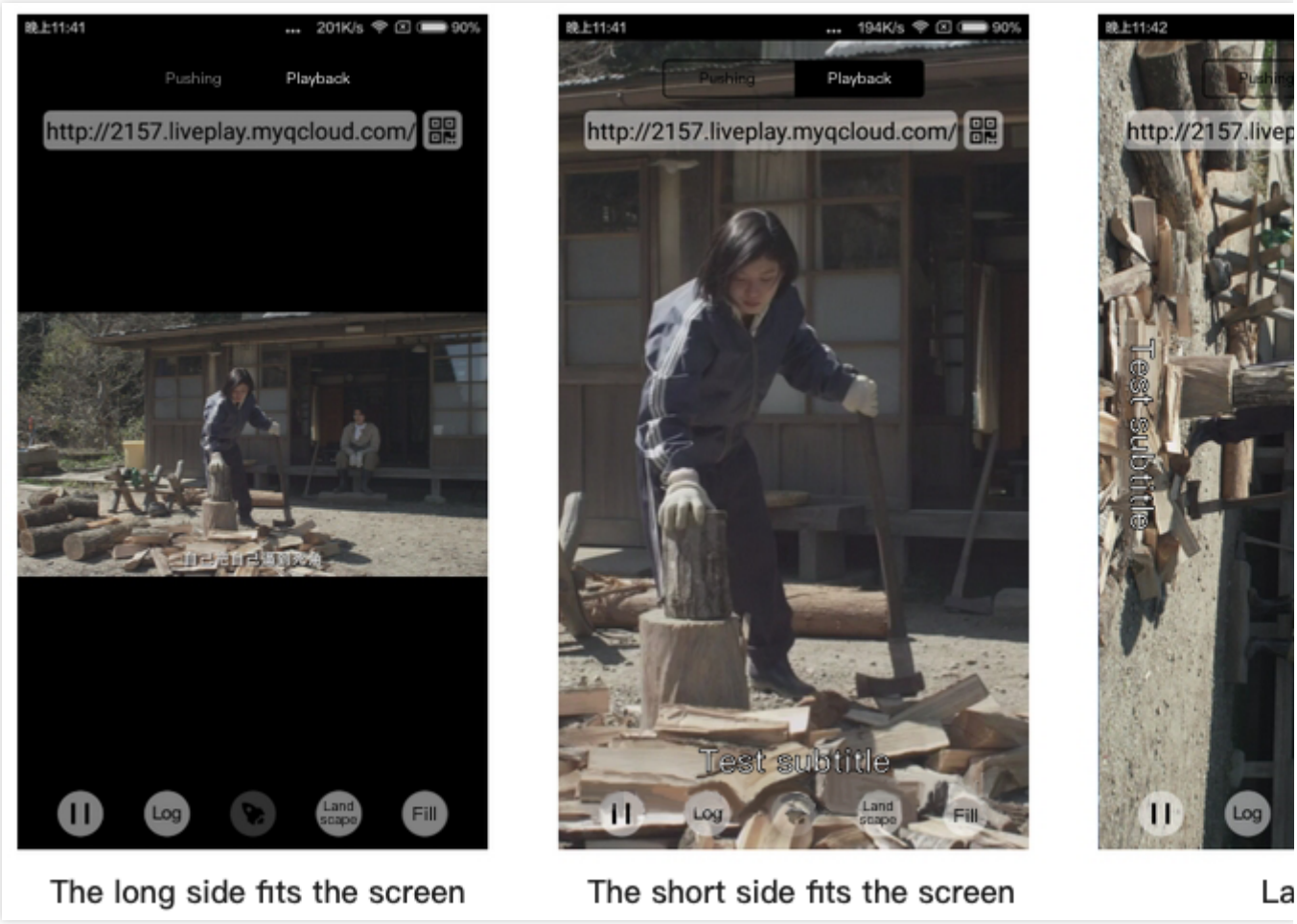
6. Adjust the image

setRenderFillMode: Fill or fit

Value	Description
V2TXLiveFillModeFill	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 entirely.
V2TXLiveFillModeFit	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.

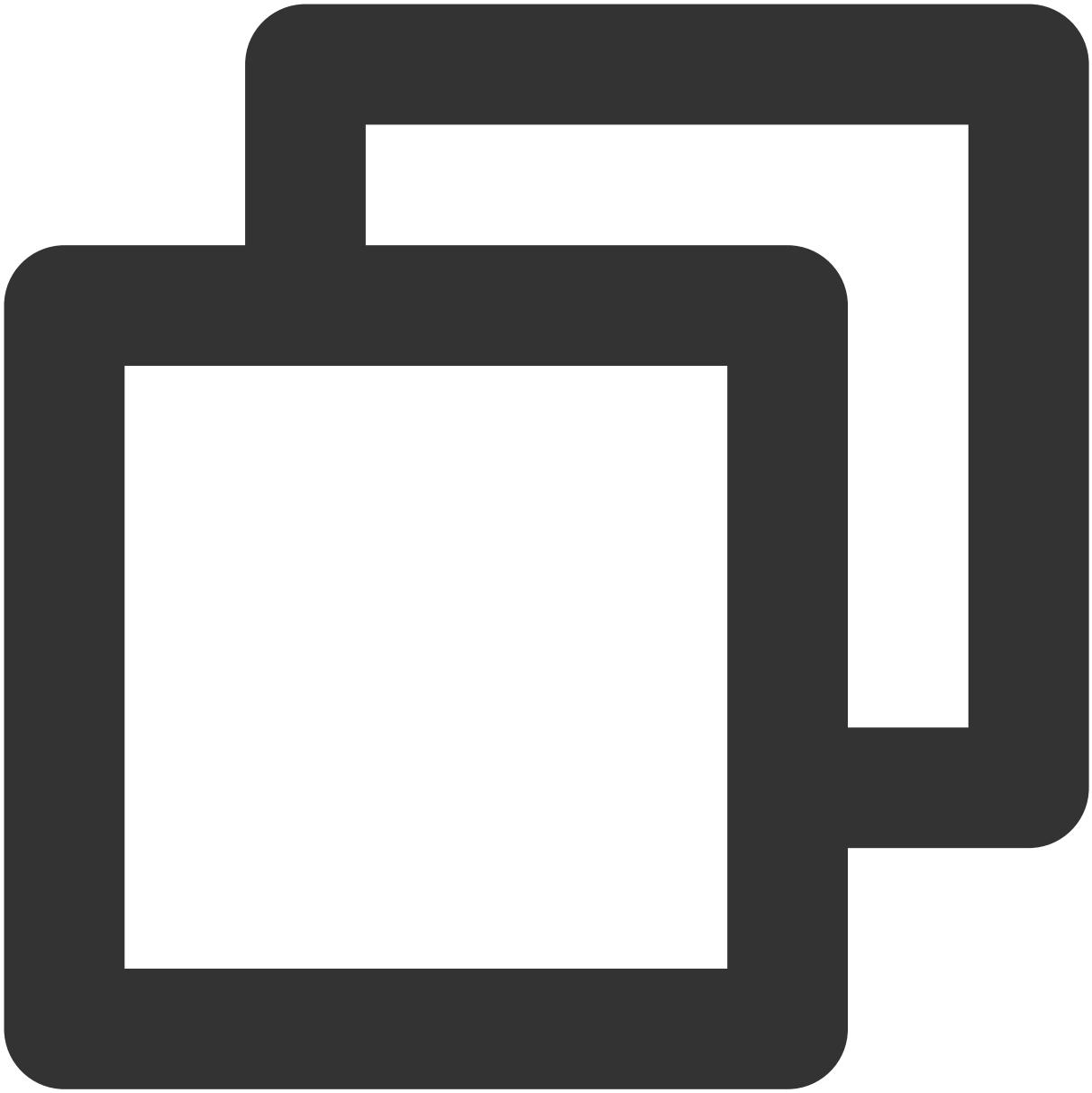
setRenderRotation: Clockwise rotation of video

Value	Description
V2TXLiveRotation0	No rotation
V2TXLiveRotation90	Rotate 90 degrees clockwise
V2TXLiveRotation180	Rotate 180 degrees
V2TXLiveRotation270	Rotate 270 degrees clockwise



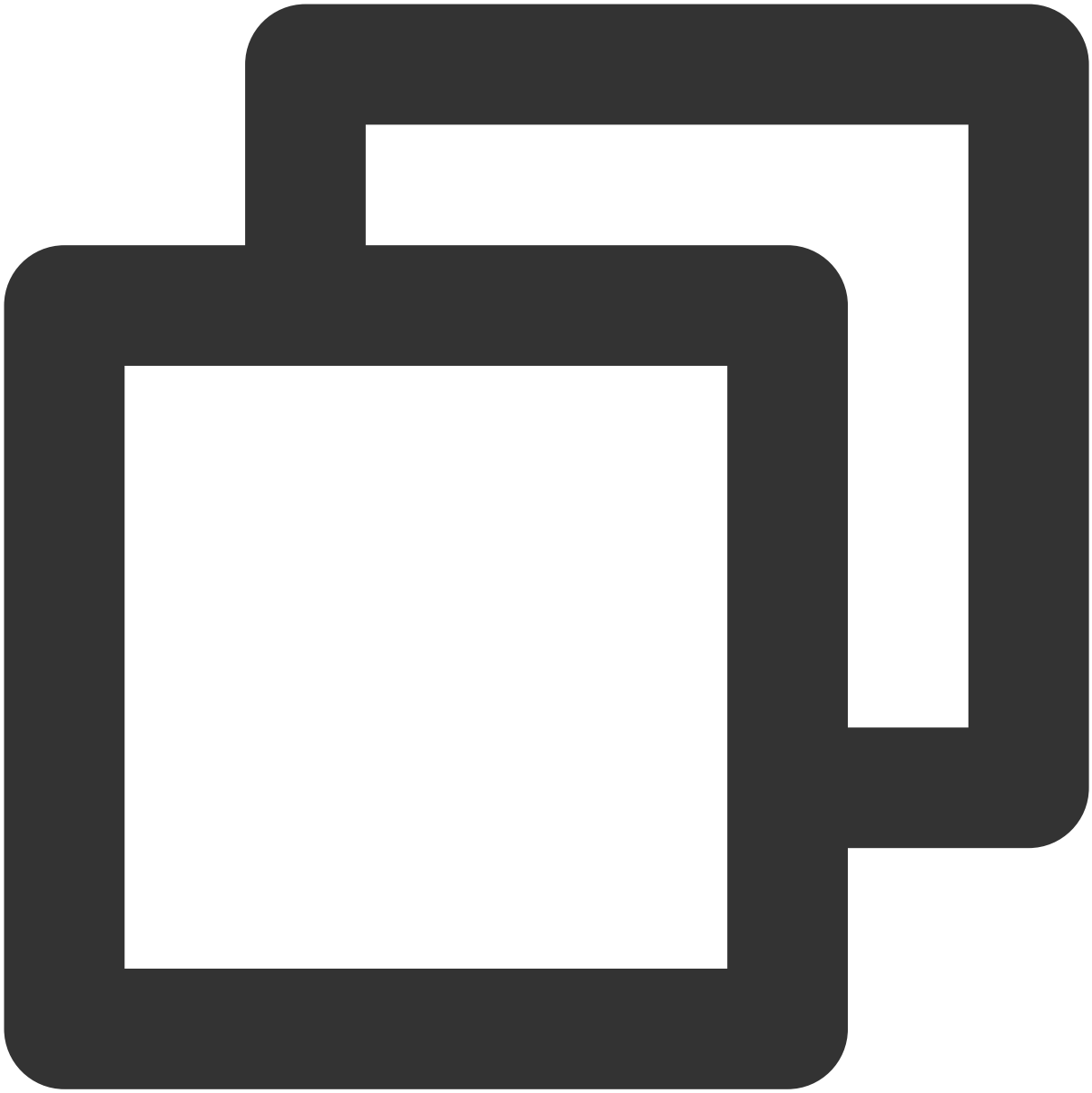
7. Pause playback

Technically speaking, you cannot pause a live playback. In this document, pausing live playback refers to **freezing the video** and **disabling the audio**. While the video is frozen, new video streams continue to be sent to the cloud. When you resume playback, it resumes playing the current latest stream. This is different from pausing and resuming playback in VOD, in which the player behaves the same way as it does when you pause and resume a local video file.



```
// Pause playback
_livePlayer.pauseAudio();
_livePlayer.pauseVideo();
// Resume playback
_livePlayer.resumeAudio();
_livePlayer.resumeVideo();
```

8. Stop playback



```
// Stop playback
_livePlayer.stopPlay();
```

Latency Control

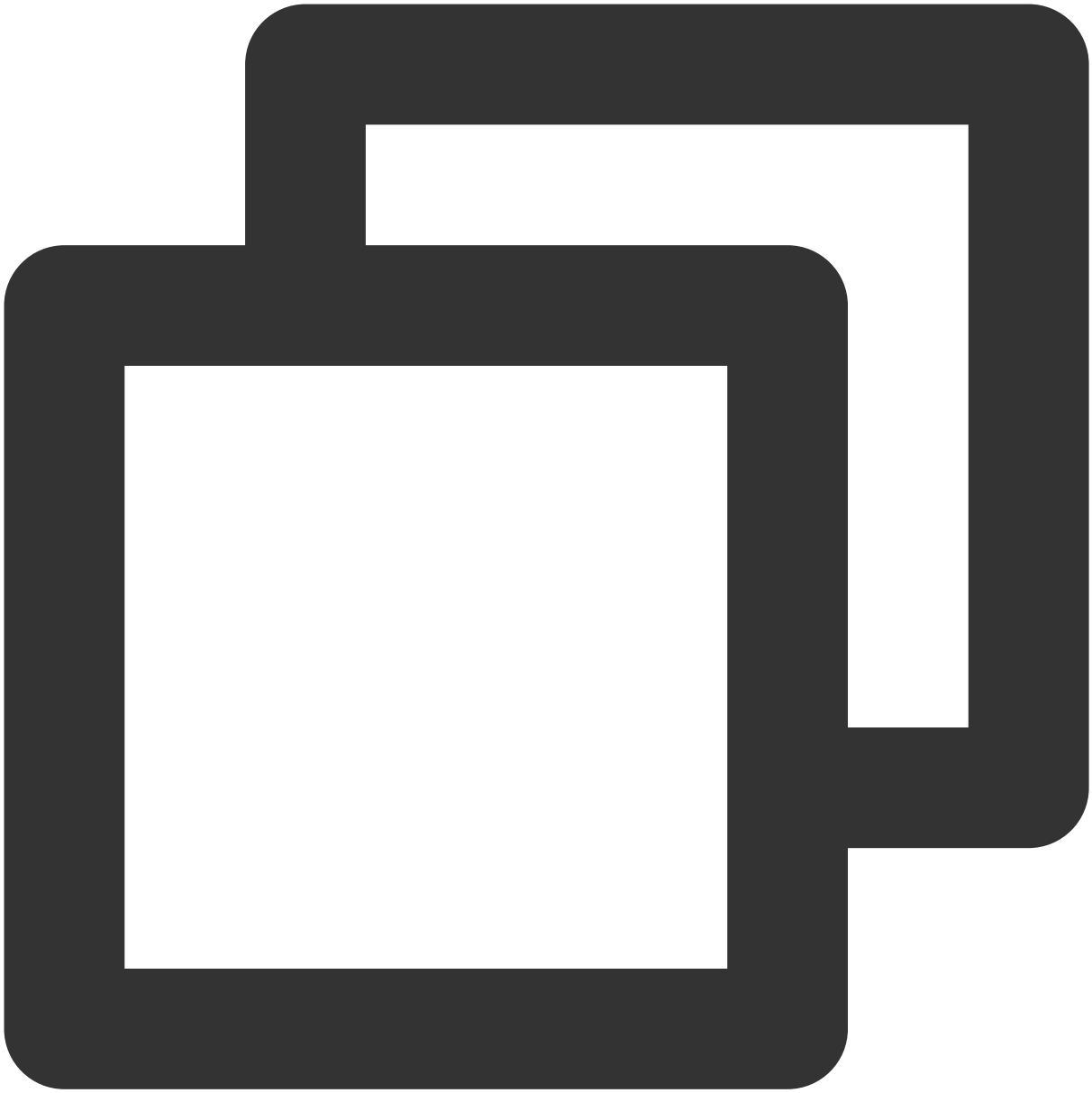
The live playback feature of the SDK is not based on FFmpeg, but Tencent Cloud's proprietary playback engine, which is why the SDK offers better latency control than open-source players do. We provide three latency control

modes, which can be used for live showrooms, game streaming, and hybrid scenarios.

Comparison of control modes

Mode	Stutter	Average Latency	Scenario	Remarks
Speedy	Relatively high	2-3s	Live showroom	Has better latency control and is suitable for scenarios that require a low latency.
Smooth	Low	>= 5s	Game streaming	Suitable for game live streaming scenarios with a high bitrate.
Auto	Adapts to network conditions	2-8s	Hybrid	The better the network conditions at the audience end, the lower the latency.

Code to integrate the three modes



```
// Auto mode
_txLivePlayer.setCacheParams(1, 5);
// Speedy mode
_txLivePlayer.setCacheParams(1, 1);
// Smooth mode
_txLivePlayer.setCacheParams(5, 5);

// Start playback after configuration
```

Note:

For more information on stutter and latency control, see Video Stutter.

Listening for SDK Events

You can bind a V2TXLivePlayerObserver to `V2TXLivePlayer`. Then, all internal SDK status information, such as player status, playback volume level, reception of the first audio/video frame, statistical data, warnings, and errors, will be notified to you through corresponding callbacks.

Periodically triggered notification

The onStatisticsUpdate notification is triggered once every 2 seconds to provide real-time feedback on the current player status. It can act as a dashboard to inform you of what is happening inside the SDK so you can better understand the current network conditions and video information.

Parameter	Description
appCpu	CPU usage (%) of the application
systemCpu	CPU usage (%) of the system
width	Video width
height	Video height
fps	Frame rate (fps)
audioBitrate	Audio bitrate (Kbps)
videoBitrate	Video bitrate (Kbps)

onPlayoutVolumeUpdate is the callback for the player volume level. It will be returned only when you call enableVolumeEvaluation to enable the prompt for the player volume level. The callback interval is the same as that specified by the `intervalMs` parameter in `enableVolumeEvaluation`.

Event-triggered notifications

Other callbacks are triggered when specific events occur.