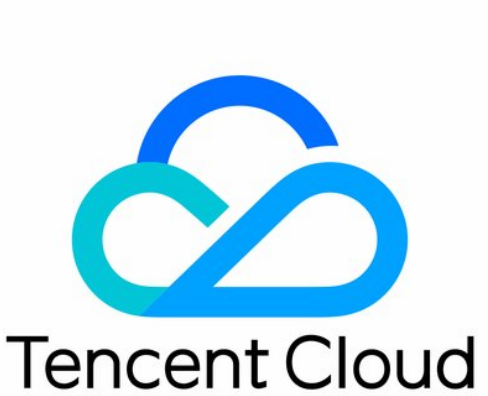
Mobile Live Video Broadcasting

WeChat Mini Program

Product Introduction



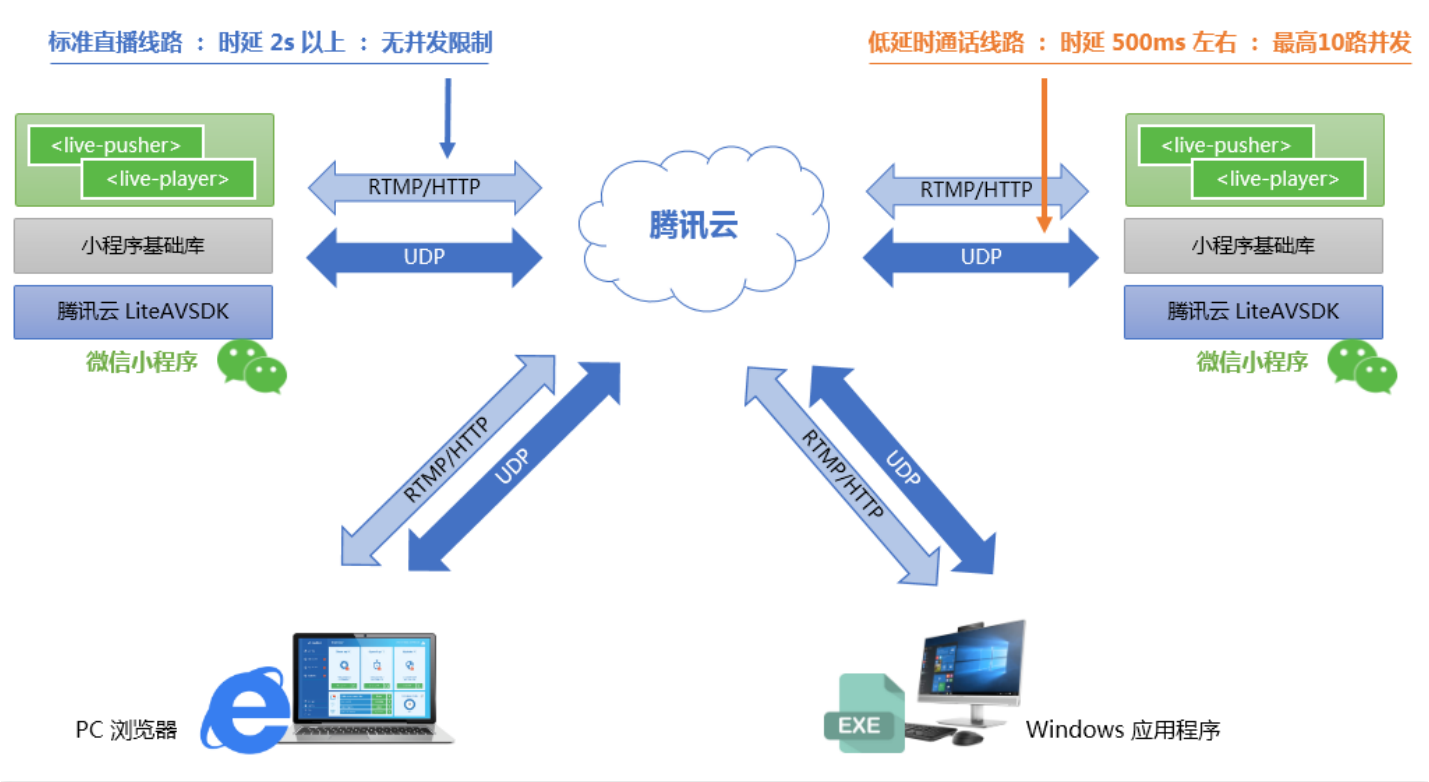
Contents

WeChat Mini Program Getting Started

Last updated：2018-10-09 16:30:45

Background

In the second half of 2017, WeChat has integrated Mobile LVB SDK into Mini Program solution, and wrap TXLivePusher and TXLivePlayer APIs of LiteAVSDK using <live-pusher> and <live-player> tags.

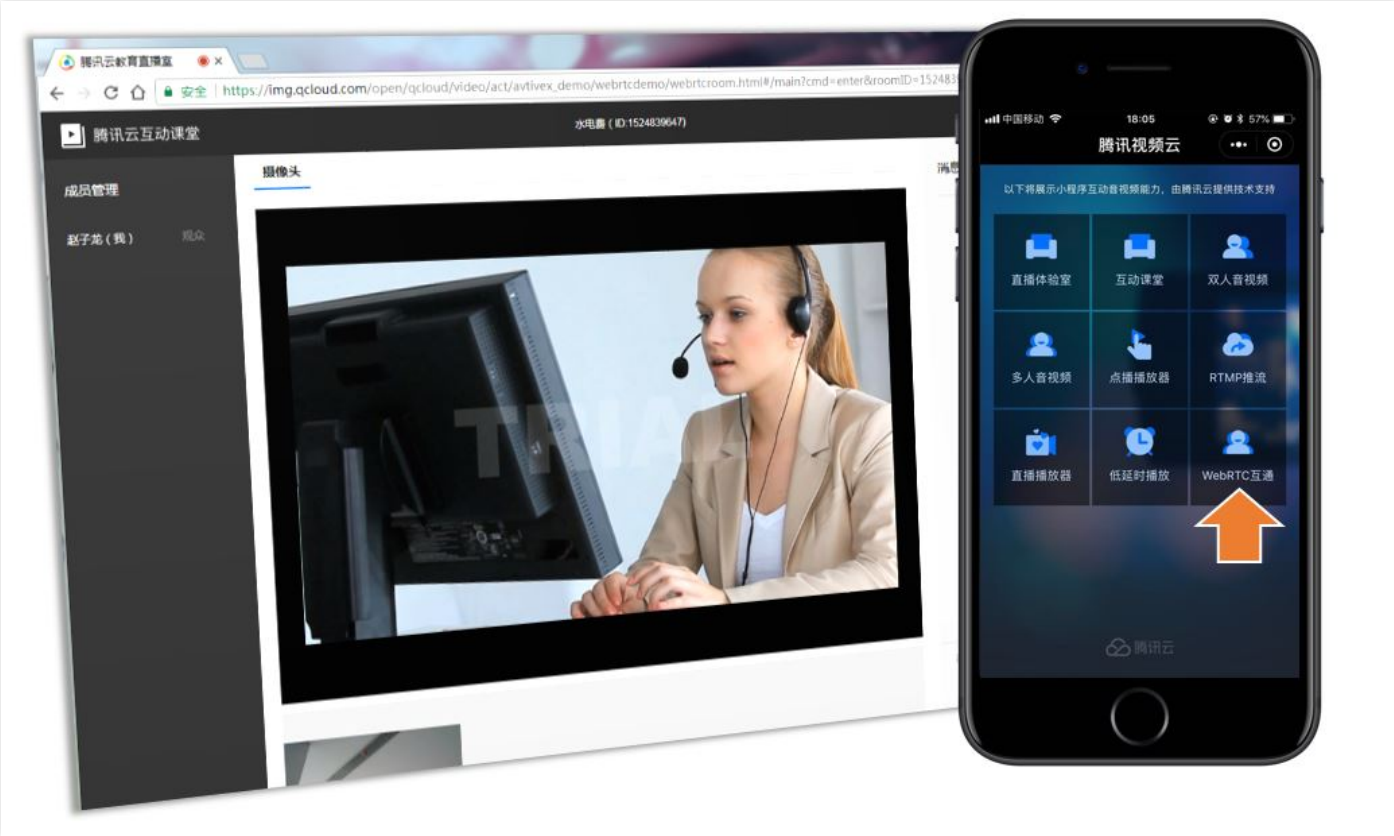


These two simple tags can implement audio/video capabilities in most application scenarios. The following describes how to use these capabilities in typical scenarios.

User Experience

- **WeChat (Mini Program)**

- **Enterprise (PC)**



Feature	Mini Program Component	Experience on PC	Dependent Cloud Service	Description
Mobile LVB	<live-room>	N/A	LVB+IM	Demonstrate personnel live video solution based on Mini Program
PC LVB	<live-room>	WebEXE	LVB+IM	Demonstrate features related to live classroom broadcasting and teacher-student interaction (in combination with PC)
One-on-one video chat	<rtc-room>	WebEXE	LVB+IM	Demonstrate one-on-one video chat feature which is applicable to online customer service
Multi-person video chat	<rtc-room>	N/A	LVB+IM	Demonstrate multi-person video chat feature which is applicable to temporary meetings

Feature	Mini Program Component	Experience on PC	Dependent Cloud Service	Description
WebRTC	<webrtc-room>	Chrome	Real-Time Audio/Video	Demonstrate the capability of interconnection between Mini Program and Chrome browser
RTMP Push	<live-pusher>	N/A	LVB	Demonstrate basic RTMP push feature
LVB Player	<live-player>	N/A	LVB	Demonstrate LVB playback feature based on RTMP and FLV protocols

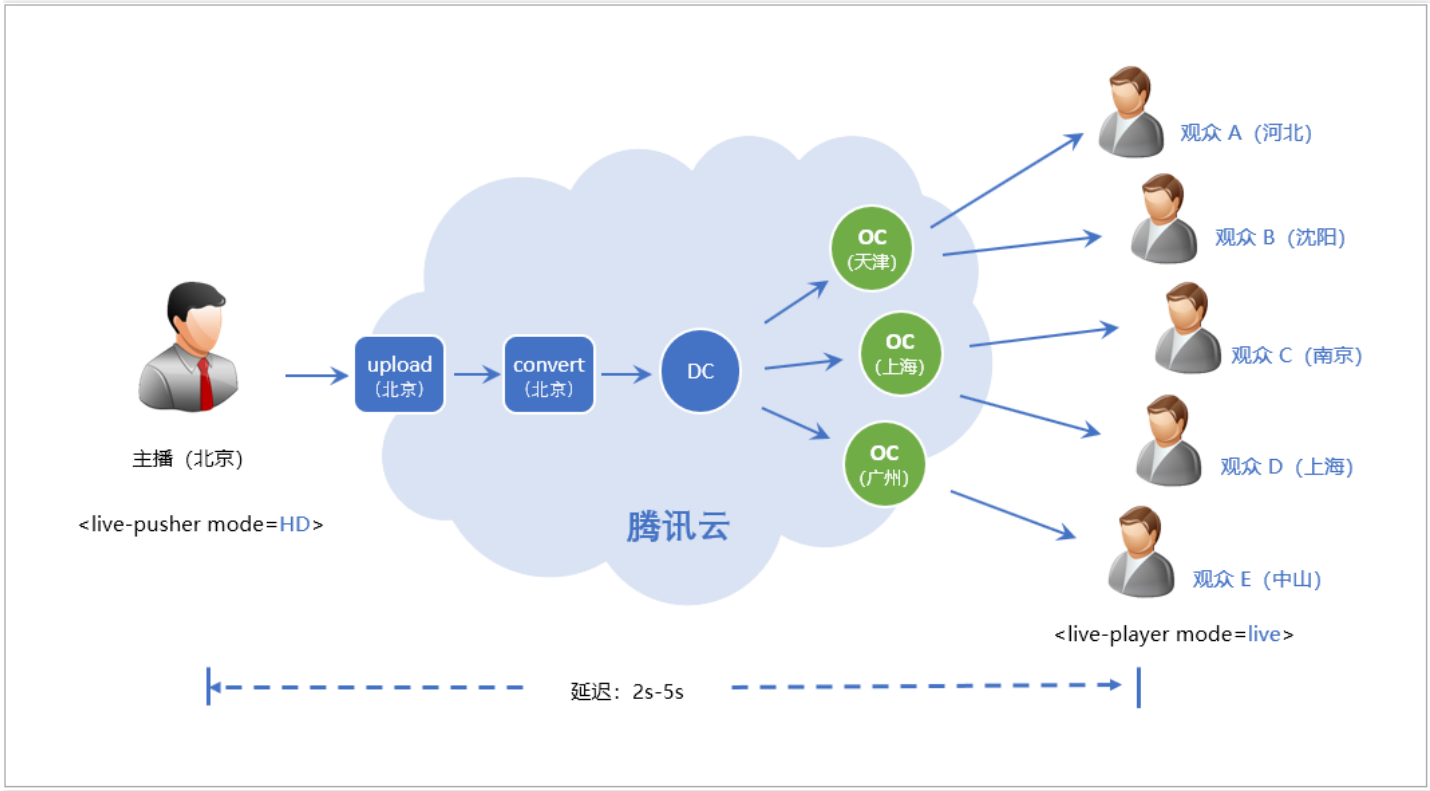
How to Enable

- **Step 1: Enable access to tags**
  For policy and compliance considerations, <live-pusher> and <live-player> are only supported for the categories in the following table.
  For mini programs meeting requirements of categories, you need to enable the access to these two tags in **Settings** -> **API Settings** of the Mini Program management backend.

Primary Category	Sub-category
"Social"	LVB
"Education"	Online education
"Healthcare"	Internet hospital and public hospital
"Government Affairs and Livelihood"	All secondary categories
"Finance"	Funds, trusts, insurance, banking, securities/futures, micro-credit of non-financial institutions, credit investigation, and consumer finance

- **Step 2: Activate Tencent Cloud LVB service**
  Audio/video capability for mini programs relies on Tencent Cloud LVB and IM services, which can be activated free of charge by clicking the link. IM service can be used immediately once activated. LVB service that has a high risk of posting pornographic and political content requires users to go through Tencent Cloud's manual audit.

Standard LVB scenario: Online Training



Introduction

Online Training is a very typical online LVB scenario, where you just need to join two tags together. <live-pusher> is used to upload local video images and audios to Tencent Cloud, while <live-player> is used to pull video streams for playback.

Tencent Cloud, as a signal amplifier, is responsible for amplifying a video stream from <live-pusher> across the country, allowing every <live-player> to pull the smooth video stream in real time from a closest CVM. This stable and reliable technique with simple principles enables millions of viewers to watch online video at the same time in case of high concurrency, and thus is the basis to implement scenarios from online education and sport events to game LVB and Huajiao.

The latency of this solution is 2 to 5 seconds on average, which can be set by using min-cache and max-cache tags of <live-player>. The smaller the value is, the lower the latency is, and therefore the higher the probability of stutter becomes.

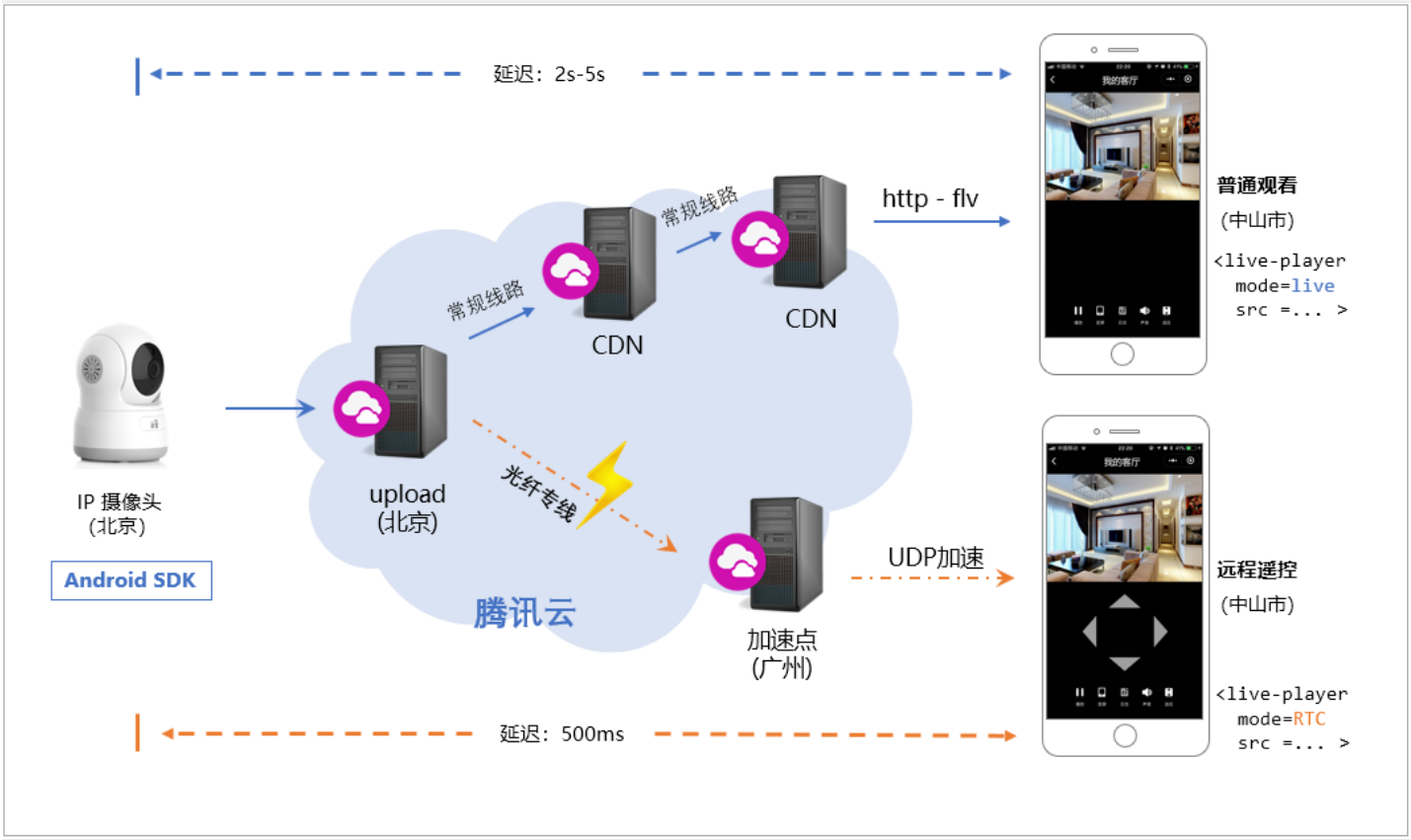
Integration

- Step 1 Obtain URL: See Quickly Obtain URL or document on how to construct URL.

- Step 2 Integrate push: Use <live-pusher> tag to push video streams to the RTMP push address (domain name: livepush.mycloud.com) obtained in step 1, and set the mode to **HD**. If any failure occurs, use the debug feature in <live-pusher> to locate problems or refer to DOC for troubleshooting.

- Step 3: Integrate playback: Use <live-player> to play RTMP or HTTP-FLV (recommended) address specified by SRC, and set the mode to **live**. "min-cache" and "max-cache" can be set to 1 and 3 respectively.

Scenario with ultra-low latency: Remote control



Introduction

In security monitoring scenario, the IPCamera for home use comes with head rotation feature, that is, the rotation of camera is remotely controlled. In case of higher latency, viewers will wait for a longer period of time to watch moving images, thus resulting in poor user experience.

Let's take online prize claw which is very popular in 2017 as another example. If the latency in remote players' video images is quite high, it is impossible to control the price claw remotely because no one can really claw a toy.

To meet this requirement, ordinary online LVB technique is no longer applicable, and we need to use ultra-low latency mode, i.e., the RTC mode of <live-player>. Meanwhile, we should also join the playback URL with a hotlink protection signature to use Tencent Cloud's ultra-low latency linkage.

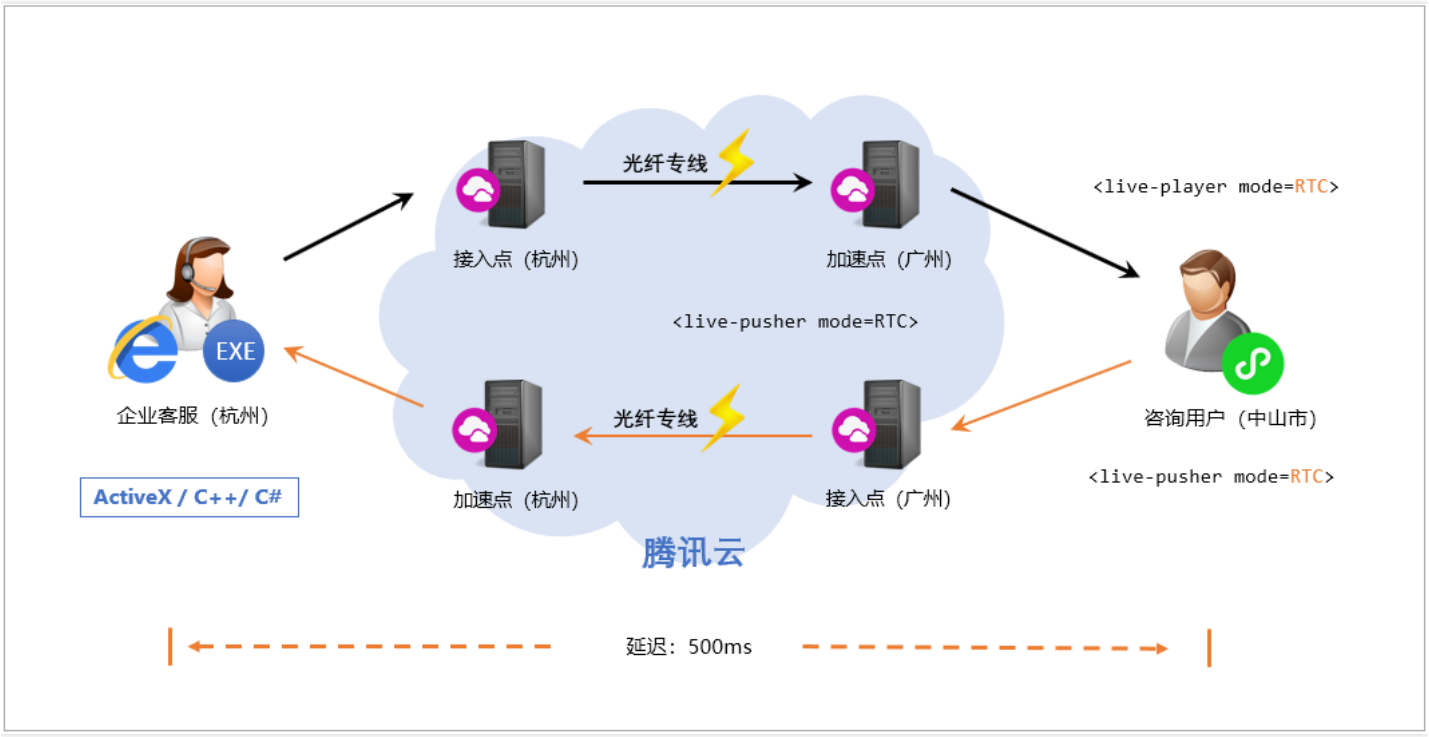
Integration

- Step 1 Obtain URL: See Quickly Obtain URL or document on how to construct URL.
- Step 2 Integrate push: You can use LiteAVSDK(TXLivePusher) for Android on your device to push streams. For other solutions, contact us by submitting a ticket or calling the number starting with 400.
- Step 3 Low latency URL: Append a Hotlink Protection Signature to an ordinary RTMP:// playback URL to convert an ordinary URL to a low latency URL, as shown below:

Item	Example	Latency
Ordinary LVB URL	rtmp://3891.liveplay.myqcloud.com/live/3891_test_clock_for_rtmpacc	>2s
Ultra-low latency URL	rtmp://3891.liveplay.myqcloud.com/live/3891_test_clock_for_rtmpacc?bizid=bizid&txTime=5FD4431C&txSerect=20e6d865f462dff61ada209d53c71cf9	<500 ms

- Step 4: RTC playback: Use the low latency URL in step 3 as the **src** attribute parameter of the <live-player> tag, and set the mode to **RTC**. In this case, **low latency control** and **UDP acceleration** features of <live-player> are enabled.

Two-way video chat: Video customer service



Introduction

In financial account opening scenario, it is critical for banks or security companies to authenticate the identity of applicants for opening an account, and audio and video recordings are also required. However, applicants are always attracted by promotions or advertisements to open an account, but it is difficult to require them to install an App before completing the entire process. Mini Program can solve this problem well. It is acceptable to users because of its formality, and thus is perfect for online financial account opening.

Likewise, insurance claim is another typical application scenario. By quickly installing and launching a mini program, the attendance of loss assessors can be reduced to save operating cost.

Integration (based on native tag)

- Step 1 Preparation: Understand how to obtain a low latency URL and how does RTC mode work by referring to the Scenario with Ultra-low Latency solution.
- Step 2 Customer service: Push a video stream to Tencent Cloud. and send the low latency playback URL A to URL B after receiving **1002** of onPushEvent. Mini program is no longer applicable to customer service, we can use IE, C++ or C# solution.
- Step 3 User: Create a <live-player> tag, set the mode to RTC, and specify SRC as URL A.

- Step 4 User: Create a <live-pusher> tag, set the mode to RTC, and send the low latency playback URL B to URL A after receiving **1002** of onPushEvent.

- Step 5 Customer service: Play low latency URL B. Mini program is no longer applicable to customer service, we can use IE, C++ or C# solution.

Integration (based on <rtc-room>)

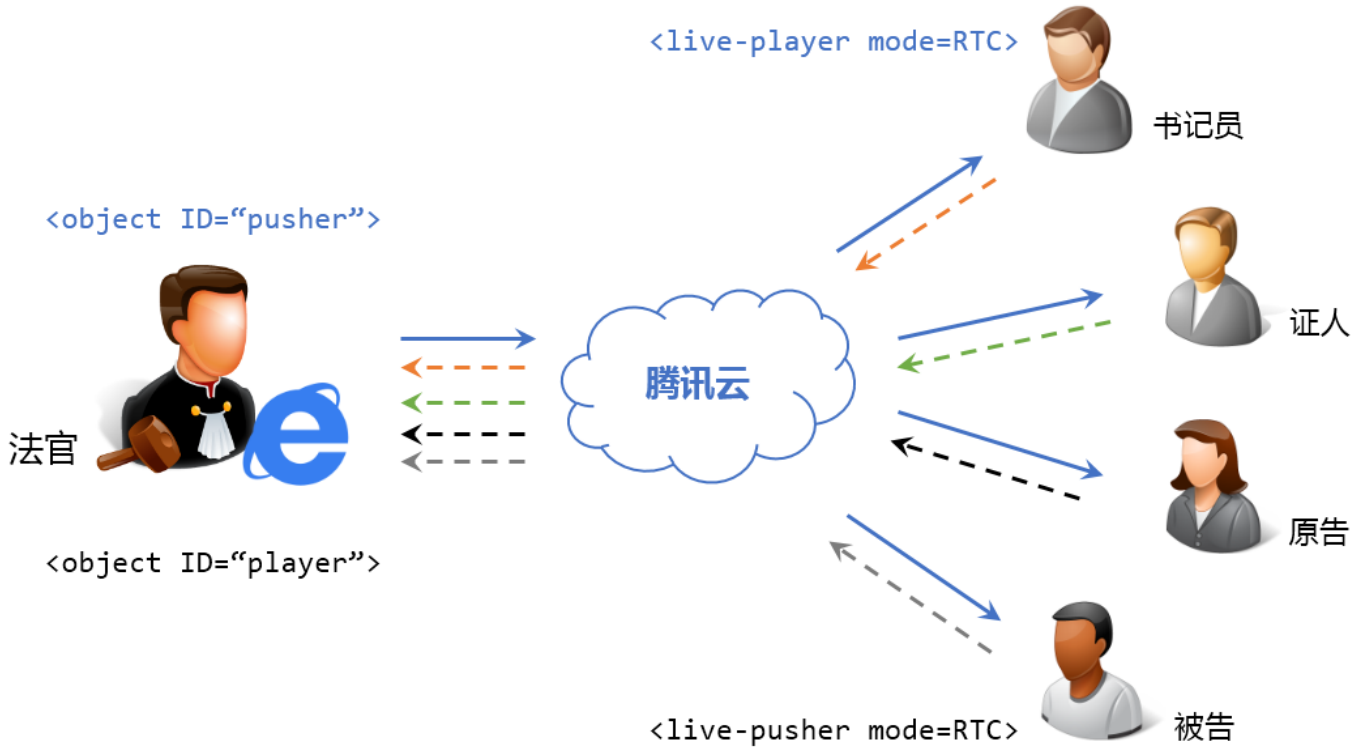
If the integration solution based on the native <live-pusher> and <live-player> is complicated, you can also implement integration based on <rtc-room> tag.

- Step 1: Activate Tencent Cloud LVB and IM services.

- Step 2 Mini Program: Use a custom component <rtc-room> to implement video chat. For 1v1 chat mode, the template of <rtc-room> can be set to **1v1**. You can also customize UI layout.

- Step 3 Windows: We provide API for multiple platforms simultaneously for you to choose based on your project needs.

Multi-person video chat: Remote hearing (RTCRoom)



Introduction

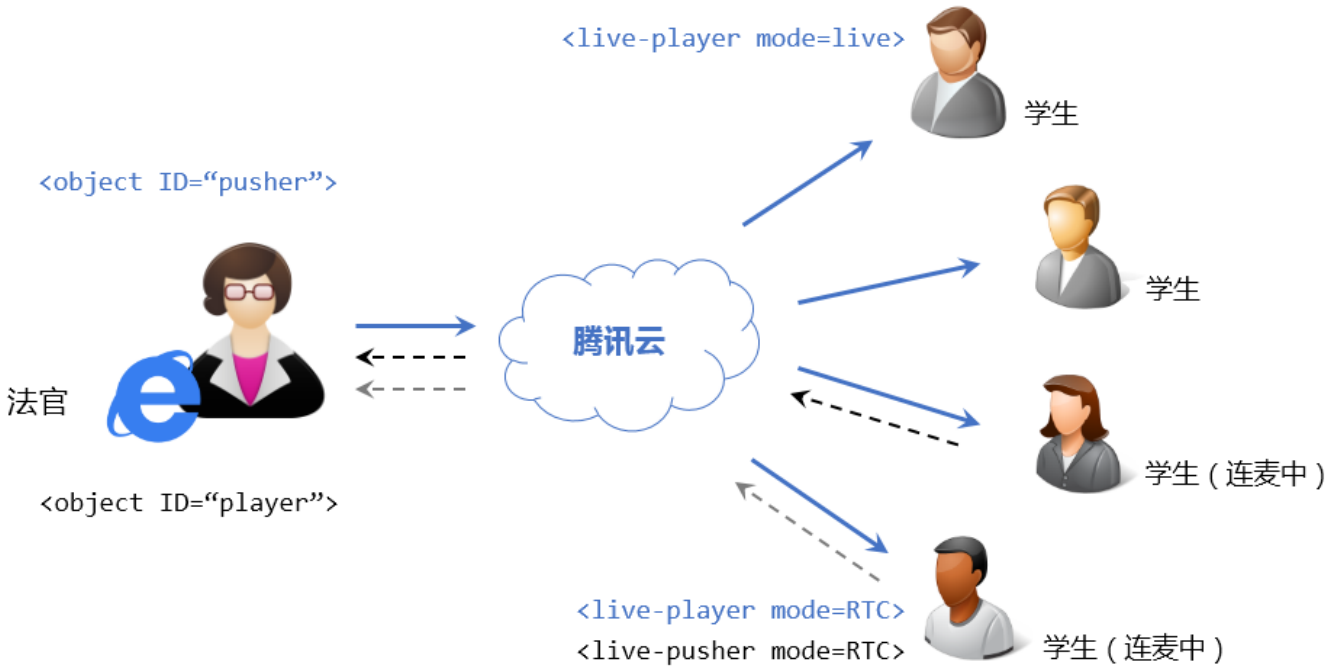
The reform of "Internet + Government affairs service" performed by Chinese government is aimed at "Provide more information and streamline procedures". Remote hearing is a typical application scenario.

The mini program of multi-person video chat is perfect for establishing a remote hearing solution. On one hand, the installation and launch of a mini program are hassle-free, making it easy for parties involved or witnesses who are inconvenient to directly appear in court to participate in hearings. On the other hand, users will feel secure with the mini program because of its formality.

Integration

- Step 1: Activate Tencent Cloud LVB and IM services.
- Step 2: Use a custom component <rtc-room> to implement video chat. You can choose one from our pre-defined modes as the template, or customize UI layout by yourself.
- Step 3 Windows: We provide API for multiple platforms simultaneously for you to choose based on your project needs.

Hybrid scenario: LVB+Joint broadcasting (LiveRoom)



Introduction

LVB scenario needs to allow millions of viewers to watch video at the same time and initiate a request for joint broadcasting to VJ, in which case hybrid scenario solution is required. Hybrid scenario is implemented by adding the joint broadcasting capability to a standard LVB scenario solution.

A common application scenario is interactive class. <rtc-room> can be directly used for small-class teaching with just a few students in a lesson. But for large-class teaching, hybrid solution must be used.

Tencent Cloud <live-room> hybrid solution comes with a separate PPT whiteboard component which is built based on the Canvas control of Mini Program.

Integration

- Step 1: Activate Tencent Cloud LVB and IM services.
- Step 2 Mini Program: Use a custom component <live-room> to implement LVB+Joint broadcasting. You can choose one from our pre-defined modes as the template, or customize UI layout by yourself.
- Step 3 Windows: We provide API for multiple platforms simultaneously for you to choose based on your project needs.

Recording

Tencent Cloud supports recording of the entire LVB session at the service end of Mini Program audio/video solution. You can enable cloud recording by following the steps below. If recording is not enabled, server nodes of Tencent Video Cloud are only responsible for transferring audio/video data without further processing.

- Step 1: Activate Tencent Cloud VOD service.
- Step 2: Log in to the LVB console (based on which the Mini Program audio/video streaming media is built), enable recording feature by going to **Access Management** -> **Access Configuration** -> **LVB Recording**. (Note: The fee here is charged by the number of concurrent recordings, not by video stream)

- Step 3: These recorded files can be found on the Video Management interface of VOD, or obtained by calling the REST API of VOD.

DEPLOYMENT GUIDE

Last updated：2018-07-09 17:49:15

View the Demo

To open the mini program demo, you need to upgrade WeChat to the latest version, go to **Discover** => **Mini Programs** => and search "Tencent Video Cloud":

Feature	Mini Program Component	Experience on PC	Dependent Cloud Service	Description
Mobile LVB	<live-room>	N/A	LVB+IM	Demonstrate personnel live video solution based on Mini Program
PC LVB	<live-room>	WebEXE	LVB+IM	Demonstrate features related to live classroom broadcasting and teacher-student interaction (in combination with PC)
One-on-one video chat	<rtc-room>	WebEXE	LVB+IM	Demonstrate one-on-one video chat feature which is applicable to online customer service
Multi-person video chat	<rtc-room>	N/A	LVB+IM	Demonstrate multi-person video chat feature which is applicable to temporary meetings
WebRTC	<webrtc-room>	Chrome	Real-Time Audio/Video	Demonstrate the capability of interconnection between Mini Program and Chrome browser
RTMP Push	<live-pusher>	N/A	LVB	Demonstrate basic RTMP push feature
LVB Player	<live-player>	N/A	LVB	Demonstrate LVB playback feature based on RTMP and FLV protocols

Register a Mini Program and Open a Relevant Interface

For policy and compliance considerations, <live-pusher> and <live-player> are not supported by all WeChat mini programs:

Mini programs of personal and enterprise accounts only support the categories in the following table:

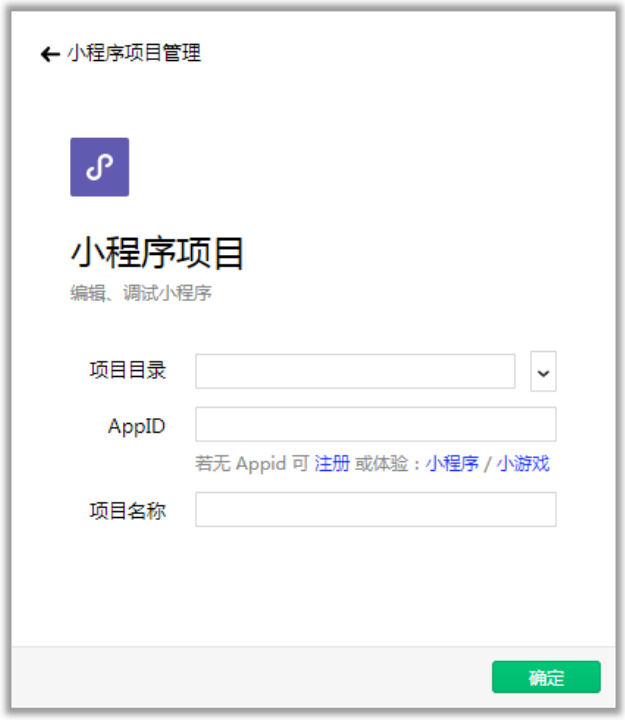
Primary Category	Sub-category
"Social"	LVB
"Education"	Online education
"Healthcare"	Internet hospital and public hospital
"Government Affairs and Livelihood"	All secondary categories
"Finance"	Funds, trusts, insurance, banking, securities/futures, micro-credit of non-financial institutions, credit investigation, and consumer finance

Open WeChat MP Platform, register and log in to the mini program, and enable the component permissions in **Settings** -> **API Settings** of the mini program management backend, as shown below:

Note: If a mini program cannot work properly while the above settings are correct. That may be because the cache within the WeChat is not updated. Delete the mini program, restart WeChat, and try again.

Install WeChat Mini Program Development Tools

Download and install the latest version of WeChat Developer Tools, and scan the QR code using the WeChat account bound to the mini program to log in to the Developer Tools.



Obtain Demo and Source Codes and Debug

- Step 1: Access SDK+Demo, and obtain the mini program demo and source codes.
- Step 2: Open WeChat Developer Tools installed, and click the **Mini Program Project** button.
- Step 3: Input the AppID of mini program, select the code directory downloaded in the previous step in the project directory (**Note:** Select the **root directory** instead of just the `wxlite` directory. The root directory contains the `project.config.json` file.), click **OK** to create a mini program project.
- Step 4: Click **OK** again to enter Developer Tools.
- Step 5: Directly scan the QR code generated by the Developer Tools to perform the test using a mobile phone.
- Step 6: Enable the debug mode to experience and debug internal features. By enabling the debug mode, you do not need to add these domain names to the whitelist of mini program.

Test Address to be Visited by Demo

The demo mini program will access the test server addresses in the following table. We offer you an experience account to access the cloud services of these servers. Usually, many customers will do tests on it. If you want to use your own backend server to avoid being disturbed by other customers, please see the following section of the document:

- **The demos related to <live-room> and <rtc-room> need to access the following addresses:**

URL	Corresponding Server Address	Server Feature Description
https://webim.tim.qq.com	IM backend server address	Used to support the messaging feature in the mini program
https://room.qcloud.com	RoomServicebackend server address	RoomService is the room management logics for supporting <rtc-room> (Video Call) and <live-room> (LVB Joint Broadcasting)

- **The demos related to <webrtc-room> need to access the following addresses:**

URL	Corresponding Server Address	Server Feature Description
`https://webim.tim.qq.com`	IM backend server address	Used to support the messaging feature in the mini program
`https://official.opensso.tencent-cloud.com/v4/openim/jsonvideoapp`	WebRTC test backend	Used to request the userSig and privateMapKey needed to enter <webrtc-room>
`https://xzb.qcloud.com/webrtc/weapp/webrtc_room`	WebRTC room list backend	A simple room list feature for Demo testing and use.

Build Your Own Account and Backend Server

In this section, we will introduce how to replace Demo's default test server address with your own server. In this way, you can use your own Tencent Cloud account to implement the above features, and it is also convenient for you to carry out secondary development.

1. Build the server of <webrtc-room>

1.1 What can this server do?

- You will see a list of rooms after clicking the interactive classroom **<webrtc-room>** feature in the demo. How is this room list implemented?

- After seeing the video room list, if you want to create a video room, or enter a video room created by others, you need to pass valid parameter values for the attributes (`sdkAppID` , `userID` , `userSig` , `roomID` and `privateMapKey`) corresponding to <webrtc-room>. How to get these several parameter values?

1.2 How to set up this server?

- Download webrtc_server, which are java source codes. The instructions in README.md will show you how to use these source codes.

1.3 How to use the built server?

- In the source codes of Mini Program, modify the `webrtcServerUrl` in the file `wxlite/config.js` to:

```
https://您自己的域名/webrtc/weapp/webrtc_room
```

- The mini program's ability to implement WebRTC is definitely for video calls with Chrome. The browser-end source codes can be downloaded by clicking Chrome(src). Modify the `serverDomain` in the file `component/WebRTCRoom.js` to:

```
https://您自己的域名/webrtc/weapp/webrtc_room
```

2. Build the server of <live-room> and <rtc-room>

2.1 What can this server do?

- Both <live-room> (for LVB joint broadcasting) and <rtc-room> (for video calls) are extensions based on Tencent Cloud's LVB and IM services and require a backend component called RoomService for running.

2.2 How to set up this server?

- Download the java source codes of RoomService. The instructions in README.md will show you how to use these source codes.

2.3 How to use the built server?

- In the source codes of Mini Program, modify the `serverUrl` and `roomServiceUrl` in the file `wxlite/config.js` to:

```
https://您自己的域名/roomservice/
```

- If the mini program uses both the <live-room> and <rtc-room> tags, it cannot be paired with the Chrome browser on PC. Instead, the WebEXE hybrid solution shall be used. Modify the `RoomServerDomain` in the files liveroom.html and double.html in the source codes of GitHub (WebEXE) to:

```
https://您自己的域名/roomservice/
```

3. Wafer zero-cost server deployment solution (Node.js)

If you are a senior web frontend engineer and you do not find the proper server for the moment, but want to have your own debugging backend quickly, you can use Tencent Cloud's Wafer feature to implement a zero-cost quick deployment solution (Wafer only supports Node.js backend codes). You only need to perform the following steps:

- Step 1: Download the source codes of mini program.

- Step 2: Complete the deployment according to Quick Deployment Guide.
- Step 3: Modify the `RoomServerDomain` in the files liveroom.html and double.html in the source codes of GitHub (WebEXE) to:

```
https://您自己的域名/roomservice/
```

WeChat-based live-pusher

Last updated：2018-07-11 11:39:39

<live-pusher> is a functional tag in mini programs to support the audio/video upstream capability. This document mainly describes how to use the tag.

Supported Version

- WeChat App for iOS: 6.5.21 and later
- WeChat App for Android: 6.5.19 and later
- Mini program basic library: 1.7.0 and later

With wx.getSystemInfo, you can obtain the version information of the current basic library.

Use Limits

For policy and compliance considerations, <live-pusher> and <live-player> are not supported by all WeChat mini programs:

- Mini programs of personal and enterprise accounts only support the categories in the following table:

Primary Category	Sub-category
"Social"	LVB
"Education"	Online education
"Healthcare"	Internet hospital and public hospital
"Government Affairs and Livelihood"	All secondary categories
"Finance"	Funds, trusts, insurance, banking, securities/futures, micro-credit of non-financial institutions, credit investigation, and consumer finance

- For mini programs meeting requirements of categories, you need to enable the component permissions in "Settings" -> "API Settings" of the mini program management backend, as shown below:

Note: If your Mini Program still does not work after the settings are correctly made, that may be because the cache within the WeChat is not updated. Delete the mini program, restart WeChat, and try again.

Attributes

Attribute Name	Type	Default Value	Description
url	String		Push URL for audio/video upstream
mode	String	RTC	SD, HD, FHD, RTC
autopush	Boolean	false	Whether to start pushing
muted	Boolean	false	Whether to mute
enable-camera	Boolean	true	Enable/disable camera
auto-focus	Boolean	true	Manual/auto focusing
orientation	String	vertical	vertical, horizontal

Attribute Name	Type	Default Value	Description
beauty	Number	0	Beauty filter level from 0 to 9. A larger value indicates a stronger effect.
whiteness	Number	0	Whiteness level from 0 to 9. A larger value indicates a stronger effect.
aspect	String	9:16	3:4, 9:16
min-bitrate	Number	200	The minimum bitrate that determines the worst image clarity
max-bitrate	Number	1,000	The maximum bitrate that determines the best image clarity
audio-quality	String	low	"low" for audio chats and "high" for HD sound
waiting-image	String		Waiting image when WeChat is switched to the backend
waiting-image-hash	String		Waiting image check value when WeChat is switched to the backend
background-mute	Boolean	false	Disable the sound collection when WeChat is running in the backend
bindstatechange	String		Specifies a JavaScript function to accept audio/video events
debug	Boolean	false	Whether to enable the debug mode

Sample Code

```
<view id='video-box'>
<live-pusher
id="pusher"
mode="RTC"
url="{{pusher.push_url}}"
autopush='true'
bindstatechange="onPush">
</live-pusher>
</view>
```

Attribute Description

- **url**
  Indicates a push URL for audio/video upstream (URL starting with "rtmp://"). For more information on how to obtain the push URL of Tencent Cloud, please see DOC.

  > The RTMP protocol in mini programs supports UDP acceleration version. Under the same network conditions, the RTMP of UDP version features better upstream speed and anti-jitter capability than the open source version.

- **mode**
  The SD, HD and FHD modes are mainly used for LVB scenarios, such as game broadcasting, online education, and remote training. SD, HD and FHD indicate three different resolution options. In these modes, mini programs take high-definition and smooth viewing experience as the top priority over low latency.

  The RTC mode is mainly used in two-way or multi-person video chats, such as financial meetings, online customer service, auto insurance loss assessment, and training sessions. In this mode, minimized point-to-point latency and high-quality sound are the top priority. In this case, high definition and smoothness of video images may be reduced.

- **orientation and aspect**
  Indicates the horizontal mode or vertical mode. Default is vertical mode. (The Home button is located directly below the image.) The aspect ratio of the video image is 3:4 or 9:16, i.e. width < height. In the horizontal mode, the aspect ratio of the video image is 4:3 or 16:9, i.e. width > height.

  The specific aspect ratio is determined by the "aspect" value. Default is 9:16 or 3:4. This is the case where the "orientation" attribute value is "vertical". If the "orientation" attribute value is "horizontal", the aspect ratio 3:4 and 9:16 is equivalent to 4:3 and 16:9 respectively.

- **min-bitrate and max-bitrate**
  Video bitrate refers to the amount of video data output by a video encoder per second. With a definite video resolution, the higher the bitrate, the more data output per second, and the better the quality of the image.

  So the two attributes of min-bitrate and max-bitrate are used to determine the minimum and maximum definition of the output video image. Due to the limitation of the network upstream, these two values

are limited. But the image clarity is a key metric for assessing the product experience, so the two attributes must hold proper values. The **"Parameter Settings"** section below describes how to set the values.

Mini programs can automatically balance the resolution and bitrate. For example, for a bitrate of 2 Mbps, mini programs select 720p resolution to match, and for a bitrate of 300 Kbps, mini programs select a lower resolution to improve the encoding efficiency. Therefore, you can control the image quality simply by the min-bitrate and max-bitrate parameters.

- **waiting-image and waiting-image-hash**
  To protect user privacy, mini programs hope to stop the camera's image collection when WeChat is running in the backend. However, this may lead to poor experience for users due to black screen or frozen screen (stay frozen on the last frame). To solve this problem, you can use the waiting-image attribute by setting an image with the meaning of "waiting" (waiting-image indicates the image URL and waiting-image-hash is the MD5 check value of the image). When WeChat is running in the backend, mini programs use the image as a substitute for the camera screen, using extremely low traffic to maintain the video stream for 3 minutes.

- **debug**
  Using the right tool is essential for debugging audio/video features. Mini programs provide the debug mode for the live-pusher tag. In the debug mode, a translucent log window is displayed on the window originally used to render the video images, which is used to display various audio/video metrics and events and reduce the difficulty of debugging. For more information on how to use it, please see FAQ.

Parameter Settings

The recommended values for these attributes are as follows:

Scenario	mode	min-bitrate	max-bitrate	audio-quality	Description
SD LVB	SD	300 Kbps	800 Kbps	high	For narrowband scenarios, such as outdoor or poor network conditions
HD LVB	HD	600 Kbps	1,200 Kbps	high	For most Apps; recommended for normal LVB scenarios
FHD LVB	FHD	600 Kbps	1,800 Kbps	high	For scenarios with high requirements for definition; HD is recommended for ordinary mobile phones users

Scenario	mode	min-bitrate	max-bitrate	audio-quality	Description
Video customer service (user)	RTC	200 Kbps	500 Kbps	high	For scenarios that take sound as the priority over images, so do not set the image quality too high
Auto insurance loss assessment (owner)	RTC	200 Kbps	1,200 Kbps	high	The image quality can be very high depending on different vehicles.
Multi-way conference (main speaker)	RTC	200 Kbps	1,000 Kbps	high	High image quality for the main speaker and low quality for participants
Multi-way conference (participants)	RTC	150 Kbps	300 Kbps	low	Do not set very high image quality and audio quality for participants

Unless very low bandwidth scenarios, do not set "low" for the audio-quality, because the audio quality and delay are much worse than those of the "high" mode.

Object Operations

- **wx.createLivePusherContext()**
  With the wx.createLivePusherContext(), the <live-pusher> tag can be associated with a JavaScript object to perform operations on the object.

- **start**
  Start push. If the "autopush" attribute of <live-pusher> is set as "false" (default value), you can start push using "start".

- **stop**
  End push.

- **pause**
  Pause push.

- **resume**
  Resume push. Use with "pause".

- **switchCamera**
  Switch between front and rear cameras.

- **snapshot**
  Push screenshot. Use the same size as the component. The temporary path for screenshots is ret.tempImagePath.

```
var pusher = wx.createLivePusherContext('pusher');
pusher.start({
success: function(ret){
console.log('start push success!')
}
fail: function(){
console.log('start push failed!')
}
complete: function(){
console.log('start push complete!')
}
});
```

Internal Events

With the **bindstatechange** attribute of the **<live-pusher>** tag, you can bind an event handling function that listens on internal events and exception notifications of push module.

1. Normal events

code	Event	Description
1001	PUSH_EVT_CONNECT_SUCC	Connected to the CVM
1002	PUSH_EVT_PUSH_BEGIN	Handshake with the server completed; everything is OK; ready to start upstream push
1003	PUSH_EVT_OPEN_CAMERA_SUCC	Camera enabled. Cannot enable the camera if the camera has been occupied or you have limited permission to the camera.

2. Critical errors

code	Event	Description
-1301	PUSH_ERR_OPEN_CAMERA_FAIL	Failed to enable the camera
-1302	PUSH_ERR_OPEN_MIC_FAIL	Failed to enable the microphone
-1303	PUSH_ERR_VIDEO_ENCODE_FAIL	Video encoding failed
-1304	PUSH_ERR_AUDIO_ENCODE_FAIL	Audio encoding failed
-1305	PUSH_ERR_UNSUPPORTED_RESOLUTION	Unsupported video resolution
-1306	PUSH_ERR_UNSUPPORTED_SAMPLERATE	Unsupported audio sampling rate
-1307	PUSH_ERR_NET_DISCONNECT	Network disconnected. Reconnection attempts have failed for three times, thus no more retries will be performed. Restart the push manually.

3. Warning events

Errors in internal warnings are recoverable. The audio/video SDKs in mini programs initiate appropriate recovery measures. The purpose of the warning is mainly to prompt developers or end users of the error, as shown below:

- **WARNING_NET_BUSY**
  The upstream network is poor. It is recommended to remind users of improving the current network environment, such as getting users closer to their routers or switching to WiFi.

- **WARNING_SERVER_DISCONNECT**
  The request is rejected by the backend. This is usually caused by miscalculated txSecret in the URL, or because the URL is occupied by others. (Unlike playback URL, only one user can use the push URL.)

code	Event	Description
1101	PUSH_WARNING_NET_BUSY	The upstream network is poor. It is recommended to remind users of improving the current network environment.
1102	PUSH_WARNING_RECONNECT	Network disconnected and auto reconnection has started (auto reconnection will be stopped after three failed attempts)

code	Event	Description
1103	PUSH_WARNING_HW_ACCELERATION_FAIL	Failed to start hardware decoding. Use software decoding instead.
1107	PUSH_WARNING_SWITCH_SWENC	Automatically switch to hardware encoding due to machine performance issues
3001	PUSH_WARNING_DNS_FAIL	DNS resolution failed and trigger retry process.
3002	PUSH_WARNING_SEVER_CONN_FAIL	Failed to connect to the server and trigger retry process.
3003	PUSH_WARNING_SHAKE_FAIL	Server handshake failed and trigger retry process.
3004	PUSH_WARNING_SERVER_DISCONNECT	Server actively disconnected and trigger retry process.
3005	PUSH_WARNING_SERVER_DISCONNECT	Socket linkage disconnected due to exception and trigger retry process.

4. Sample codes

```
Page({
onPush: function(ret) {
if(ret.detail.code == 1002) {
console.log('Push is successful',ret);
}
},

/**
* Lifecycle function - listening page loading
*/
onLoad: function (options) {
//...
}
})
```

live-player

Last updated：2018-07-11 11:40:27

≪live-player> is a functional tag in mini programs to support the audio/video downstream (playback) capability. This document mainly describes how to use the tag.

Supported Version

- WeChat App for iOS: 6.5.21 and later
- WeChat App for Android: 6.5.19 and later
- Mini program basic library: 1.7.0 and later

With wx.getSystemInfo, you can obtain the version information of the current basic library.

Use Limits

For policy and compliance considerations, <live-pusher> and <live-player> are not supported by all WeChat mini programs:

- Mini programs of personal and enterprise accounts only support the categories in the following table:

Primary Category	Sub-category
"Social"	LVB
"Education"	Online education
"Healthcare"	Internet hospital and public hospital
"Government Affairs and Livelihood"	All secondary categories
"Finance"	Funds, trusts, insurance, banking, securities/futures, micro-credit of non-financial institutions, credit investigation, and consumer finance

- For mini programs meeting requirements of categories, you need to enable the component permissions in "Settings" -> "API Settings" of the mini program management backend, as shown below:

Note: If your Mini Program still does not work after the settings are correctly made, that may be because the cache within the WeChat is not updated. Delete the mini program, restart WeChat, and try again.

Attributes

Attribute Name	Type	Default Value	Description
src	String		Playback URL for audio/video downstream, which can be in the RTMP and FLV format
mode	String	live	live, RTC
autoplay	Boolean	false	Whether to play automatically
muted	Boolean	false	Whether to mute
orientation	String	vertical	vertical, horizontal
object-fit	String	contain	contain, fillCrop
background-mute	Boolean	false	Turn off the sound when WeChat is running in the backend
min-cache	Number	1	Minimum buffer delay (in sec)

Attribute Name	Type	Default Value	Description
max-cache	Number	3	Maximum buffer delay (in sec)
bindstatechange	EventHandler		Specifies a JavaScript function to accept the player event
bindfullscreenchange	EventHandler		Specifies a JavaScript function to accept the full screen event
debug	Boolean	false	Whether to enable the debug mode

Sample Code

```
<view id='video-box'>
<live-player
wx:for="{{player}}"
id="player_{{index}}"
mode="RTC"
object-fit="fillCrop"
src="{{item.playUrl}}"
autoplay='true'
bindstatechange="onPlay">
</live-player>
</view>
```

Ultra-low Latency

The RTC mode of <live-player> supports ultra-low latency linkages within 500 ms. It is suitable for such scenarios as video chats and remote control. To use ultra-low latency mode during the playback, the following points should be taken into consideration:
(1) If the push end is a WeChat Mini Program, use the RTC mode of <live-pusher>.
(2) If the push end is an iOS or Android SDK, use the MAIN_PUBLISHER mode of setVideoQuality.
(3) If the push end is Windows, do not use OBS due to the high latency. The Windows SDK is recommended.
(4) Do not change the min-cache and max-cache values of <live-player>. Use the default values.
(5) Use the ultra-low latency playback URL, i.e. a URL starting with "rtmp://" with a hotlink protection signature, as shown below:

Item	Example	Latency
Ordinary LVB URL	rtmp://3891.liveplay.myqcloud.com/live/3891_test_clock_for_rtmpacc	>2s
Ultra-low latency URL	rtmp://3891.liveplay.myqcloud.com/live/3891_test_clock_for_rtmpacc?bizid=bizid&txTime=5FD4431C&txSerect=20e6d865f462dff61ada209d53c71cf9	< 500ms

Attribute Description

- **src**
  Indicates a playback URL for audio/video downstream, which can be in the RTMP format (URLs starting with "rtmp://") and FLV format (URLs starting with "http://" and ending with ".flv"). For more information on how to obtain the push URL of Tencent Cloud, please see DOC.

  > The HLS (m3u8) protocol can be used in <video>, so the ≪live-player> tag does not support HLS (m3u8). However, HLS (m3u8) protocol is not recommended for live videos. The delay using HLS will be 10 times higher than that using RTMP and FLV protocols.

- **mode**
  The live mode is mainly used for LVB scenarios, such as game broadcasting, online education, and remote training. In this mode, the modules in mini programs will take smooth viewing experience as the priority. By setting the min-cache and max-cache attributes, you can adjust the latency experienced by viewers (playback). The following sections will further describe these two parameters.

  The RTC mode is mainly used in two-way or multi-person video chats, such as financial meetings, online customer service, auto insurance loss assessment, and training sessions. In this mode, the settings for min-cache and max-cache will not take effect, because the delay will be automatically controlled at a very low level (about 500 ms) in mini programs.

- **min-cache and max-cache**
  The two parameters are used to specify the minimum and maximum buffer time for viewers. The **buffer time** works like a "reservoir", by which players can ease the impact of network fluctuations on viewing smoothness. When stutter or freezing occurs in network packets, the "emergency water" in the "reservoir"

can keep players working for a short period of time. As long as the private network speed becomes normal again in this period, smooth video images are rendered continuously in players.

The more "water" in the "reservoir", the better capability to resist the network fluctuations, and the higher latency for viewers as a consequence. Therefore, it is necessary to use different configurations in different scenarios to achieve the optimal experience balance.

- For LVB streams with a relatively low bitrate (about 1 Mbps and mainly for character images), min-cache = 1 and max-cache = 3 are recommended.
- For LVB streams with a relatively high bitrate (about 2-3 Mbps and mainly for HD game images), min-cache = 3 and max-cache = 5 are recommended.

In the RTC mode, the settings for the two parameters do not take effect.

- **orientation**
  Indicates the image rendering orientation. horizontal: original image direction; vertical: 90° clockwise rotated.

- **object-fit**
  Indicates the image filling mode. contain: Keep the original aspect ratio of an image, but black edges may appear if the aspect ratio of the video image does not match that of the <live-player> tag. fillCrop: Fill the screen, but the extra part of an image may be cut out if the aspect ratio of the video image does not match that of the <live-player> tag.

- **background-mute**
  Indicates whether to turn off the sound when WeChat is running in the backend, to avoid the impact on the video that is being played in mini programs due to screen locking.

- **sound-mode**
  Specifies the playback mode, which can be: ear or speaker. ear: Play sound through a receiver; speaker: Play sound through a speaker. Default is speaker.

- **debug**
  Using the right tool is essential for debugging audio/video features. Mini programs provide the debug mode for the live-pusher tag. In the debug mode, a translucent log window is displayed on the window originally used to render the video images, which is used to display various audio/video metrics and events and reduce the difficulty of debugging. For more information on how to use it, please see FAQ.

Object Operations

- **wx.createLivePlayerContext()**
  With the wx.createLivePlayerContext(), the <live-player> tag can be associated with a JavaScript object to perform operations on the object.

- **play**
  Start playback. If the "autoplay" attribute of <live-player> is set as "false" (default value), you can start playback using "play".

- **stop**
  Stop playback.

- **pause**
  Pause playback and freeze the last frame.

- **resume**
  Resume playback. Use with "pause".

- **mute**
  Turn off the sound.

- **requestFullScreen**
  Enter full screen.

- **exitFullScreen**
  Exit full screen.

```
var player = wx.createLivePlayerContext('pusher');
player.requestFullScreen({
success: function(){
console.log('enter full screen mode success!')
}
fail: function(){
console.log('enter full screen mode failed!')
}
complete: function(){
console.log('enter full screen mode complete!')
}
});
```

Internal Events

With the **bindstatechange** attribute of the **<live-player>** tag, you can bind an event handling function that listens on internal events and exception notifications of push module.

1. Key events

code	Event	Description
2001	PLAY_EVT_CONNECT_SUCC	Connected to the CVM
2002	PLAY_EVT_RTMP_STREAM_BEGIN	The server is transmitting audio/video data.
2003	PLAY_EVT_RCV_FIRST_I_FRAME	The network receives the first piece of audio/video data.
2004	PLAY_EVT_PLAY_BEGIN	Video playback starts. Before this event, you can use the default picture to represent the waiting status.
2006	PLAY_EVT_PLAY_END	Video playback ends
2007	PLAY_EVT_PLAY_LOADING	In the loading status. In this case, the player is waiting for or collecting data from the server.
-2301	PLAY_ERR_NET_DISCONNECT	Network disconnected and unable to reconnect. The player has stopped playing.

When viewers is playing videos from a FLV URL starting with "HTTP://", mini programs will not throw the PLAY_EVT_PLAY_END event in case of disconnection of the LVB stream. This is because the FLV protocol does not define a stop event. As an alternative, you can monitor the PLAY_ERR_NET_DISCONNECT event.

2. Warning Events

Errors in internal warnings are recoverable. The audio/video SDKs in mini programs initiate appropriate recovery measures. The purpose of the warning is mainly to prompt developers or end users of the error, as shown below:

code	Event	Description
2101	PLAY_WARNING_VIDEO_DECODE_FAIL	Failed to decode the current video frame

code	Event	Description
2102	PLAY_WARNING_AUDIO_DECODE_FAIL	Failed to decode the current audio frame
2103	PLAY_WARNING_RECONNECT	Network disconnected and auto reconnection has started (the PLAY_ERR_NET_DISCONNECT event will be thrown after three failed attempts)
2104	PLAY_WARNING_RECV_DATA_LAG	Video stream is not stable. This may be caused by bad network connection.
2105	PLAY_WARNING_VIDEO_PLAY_LAG	Stutter occurred during the current video playback
2106	PLAY_WARNING_HW_ACCELERATION_FAIL	Failed to start hardware decoding. Use software decoding instead.
2107	PLAY_WARNING_VIDEO_DISCONTINUITY	Current video frames are discontinuous. This may be caused by frame loss. Blurred screen may occur.
3001	PLAY_WARNING_DNS_FAIL	DNS resolution failed (thrown only if the playback URL starts with "RTMP://")
3002	PLAY_WARNING_SEVER_CONN_FAIL	Server connection failed (thrown only if the playback URL starts with "RTMP://")
3003	PLAY_WARNING_SHAKE_FAIL	Server handshake failed (thrown only if the playback URL starts with "RTMP://")

3. Sample codes

```
Page({
onPlay: function(ret) {
if(ret.detail.code == 2004) {
console.log('Video playback starts',ret);
}
},

/**
* Lifecycle function - listening page loading
*/
onLoad: function (options) {
//...
}
})
```

Notes

1. <live-player> is a native component created by a client, whose level is the highest. You can use <cover-view> and <cover-image> to overwrite it.

2. Do not use <live-player> in <scroll-view>.

rtc-room

Last updated：2018-07-11 11:40:40

Tag Overview

The **<rtc-room>** tag implemented based on <live-pusher> and <live-player> is a custom component used for audio/video chats among two or more people. It is mainly suitable for many-to-many audio/video chat scenarios (unlike <live-room>).

Demo

You can scan the following QR code or search **Tencent Video Cloud** in WeChat Mini Program to open the demo mini program. The **two-person audio/video chats** and **multi-person audio/video chats** in the demo is the typical application scenario of <rtc-room>.

Tag Description

Attributes

Attribute	Type	Value	Description
template	String	'1v3','1v1'	(Required) Identifies the UI template used by components (If you need to customize the UI, please see UI Customization)
roomID	String	''	(Optional) Room ID
roomName	String	''	(Optional) Room name
beauty	Number	0-5	(Optional) Beauty filter level: 0-5. Default is 5
muted	Boolean	true, false	(Optional) Whether to mute. Default is false
debug	Boolean	true, false	(Optional) Whether to print push debug information. Default is false
bindonRoomEvent	function		(Required) Listens the events returned by <rtc-room> component

APIs

The **<rtc-room>** component contains the following APIs. You need to obtain the reference of the <rtc-room> tag using selectComponent before performing appropriate operations.

Function Name	Description
start()	Start
pause()	Pause
resume()	Resume
stop()	Stop
switchCamera()	Switch camera
sendTextMsg(text:String)	Send text messages

```
var rtcroom = this.selectComponent("#rtcroomid")
rtcroom.pause();
```

Event notification

The **<rtc-room>** tag returns internal events through **onRoomEvent**. The format of the event parameters is as follows:

```
"detail": {
"tag": "The unique event tag ID",
"code": "Event code",
"detail": "Detailed parameters of the event"
}
```

Sample Code

```
//Page.wxml file
<rtc-room id="rtcroomid"
roomID="{{roomID}}"
roomName="{{roomName}}"
template="1v3"
beauty="{{beauty}}"
muted="{{muted}}"
debug="{{debug}}"
bindonRoomEvent="onRoomEvent">
</rtc-room>


//Page.js file
Page({
data: {
//...
roomID: '',
roomName: '',
beauty: 3,
muted: false,
debug: false,
},
//...
onRoomEvent: function(e){
switch(e.detail.tag){
case 'roomClosed': {
//Room is closed
break;
}
case 'error': {
//An error occurred
var code = e.detail.code;
var detail = e.detail.detail;
```

```
break;
}
case 'recvTextMsg': {
//Receive a text message
var text = e.detail.detail;
break;
}
}
},

onShow: function () {
},

onHide: function () {
},

onRead: function() {
var rtcroom = this.selectComponent("#rtcroomid")
this.setData({
roomName: 'Test',
});
rtcroom.start();
},

})
```

Usage Guide

Step 1: Activate appropriate cloud service

Audio/video capability for mini programs relies on Tencent Cloud LVB and IM services, which can be activated free of charge by clicking the link. IM service can be used immediately once activated. LVB service that has a high risk of posting pornographic and political content requires users to go through Tencent Cloud's manual audit.

Step 2: Download custom component source code

<rtc-room> is not a native tag provided by WeChat Mini Program, but a custom component. Therefore, you need additional code to support this tag. Click Mini Program Source Code to download the source code package, and you can obtain the required files under the `wxlite` folder.

Step 3: Log in to RoomService (Required)

Before using the **<rtc-room>** tag, log in first by calling `/utils/rtcroom.js` to connect to RoomService in the backend.

```
var rtcroom = require('/utils/rtcroom.js');
...
rtcroom.login({
serverDomain: '',
userID: '',
userSig: '',
sdkAppID: '',
accType: '',
userName: '' //Custom user nickname
});
```

For more information about how to enter parameters, please see DOC.

Step 4: Get room lists (optional)

If you want to use the existing room list of RoomService, rather than implementing a new one, you can obtain the list information by calling the `getRoomList` function of `/utils/rtcroom.js`.

```
var rtcroom = require('/utils/rtcroom.js');
...
rtcroom.getRoomList({
data: {
index: 0, //List index number
cnt: 20 //Number of lists to be pulled
},
success: function (ret) {
console.log('Get room lists:', ret.rooms);
},
fail: function (ret) {
console.error('Failed to get room lists :', ret);
}
});
```

Step 5: Introduce the component into the project

- Reference the component in the JSON configuration file under the page directory. This is required because <rtc-room> is not a native tag.

```
"usingComponents": {
"rtc-room": "/pages/rtc-room/rtc-room"
}
```

- Use the tag in the wxml file under the page directory.

```
<rtc-room id="rtcroomid"
roomID="{{roomID}}"
roomName="{{roomName}}"
template="1v3"
beauty="{{beauty}}"
muted="{{muted}}"
debug="{{debug}}"
bindonRoomEvent="onRoomEvent">
</rtc-room>
```

Step 6: Create a room

- If you want to use the roomID assigned by RoomService, specify a roomName for <rtc-room>.

```
//Create a room (a roomID assigned by RoomService)
this.setData({
roomName: 'Test'
});
rtcroom.start();
```

- If you want to specify a roomID, set the roomID attribute before calling the start() function.

```
//Create a room (a roomID that you specify)
this.setData({
roomID: 12345
});
rtcroom.start();
```

- In either case, a room is created only when the **start()** function is called.

Step 7: Join a room

- If a room with a specified roomID has already been created, calling start() will enter the created room, rather than creating a new one.

```
this.setData({
roomID: 12345
});
rtcroom.start();
```

UI customization

If the default "1v1" and "1v3" screen layouts do not suite your needs, you can customize the screen based on the specific project:

- **Create a UI template**

```
//Step 1: Create a /pages/templates/mytemplate folder and mytemplate.wxml and mytemplate.wxss fil
es.

//Step 2: Write the mytemplate.wxml and mytemplate.wxss files.
//mytemplate.wxml
<template name='mytemplate'>
<view class='videoview'>
<view class="pusher-box">
<live-pusher
id="rtcpusher"
autopush
mode="RTC"
url="{{pushURL}}"
aspect="{{aspect}}"
min-bitrate="{{minBitrate}}"
max-bitrate="{{maxBitrate}}"
audio-quality="high"
beauty="{{beauty}}"
muted="{{muted}}"
waiting-image="https://mc.qcloudimg.com/static/img/
daeed8616ac5df256c0591c22a65c4d3/pause_publish.jpg"
background-mute="{{true}}"
debug="{{debug}}"
bindstatechange="onPush"
binderror="onError">
<cover-image class='character' src="/pages/Resources/mask.png"></cover-image>
<cover-view class='character' style='padding: 0 5px;'>Me</cover-view>
</live-pusher>
</view>
<view class="player-box" wx:for="{{members}}" wx:key="userID">
<view class='poster'>
<cover-image class='set'
src="https://miniprogram-1252463788.file.myqcloud.com/roomset_{{index + 2}}.png">
</cover-image>
</view>
```

```
<live-player
id="{{item.userID}}"
autoplay
mode="RTC"
wx:if="{{item.accelerateURL}}"
object-fit="fillCrop"
min-cache="0.1"
max-cache="0.3"
src="{{item.accelerateURL}}"
debug="{{debug}}"
background-mute="{{true}}"
bindstatechange="onPlay">
<cover-view class='loading' wx:if="{{item.loading}}">
<cover-image src="/pages/Resources/loading_image0.png"></cover-image>
</cover-view>
<cover-image class='character' src="/pages/Resources/mask.png"></cover-image>
<cover-view class='character' style='padding: 0 5px;'>{{item.userName}}</cover-view>
</live-player>
</view>
</view>
</template>

//mytemplate.wxss
.videoview {
background-repeat:no-repeat;
background-size: cover;
width: 100%;
height: 100%;
}
```

- **Introduce a template into the rtc-room component**

```
//Add a custom template for the liveroom.wxml file in the <rtc-room> component
<import src='/pages/templates/mytemplate/mytemplate.wxml'/>
<view class='conponent-box'>
<view styles="width:100%;height=100%;" wx:if="{{template=='1v3' || template=='1v1'}}">
<template is='mytemplate' data="{{pushURL, aspect,
minBitrate, maxBitrate, beauty, muted, debug, members}}"/>
</view>
</view>

//Add a custom style for the liveroom.wxss file in the <rtc-room> component
@import "../templates/mytemplate/mytemplate.wxss";
```

Other platforms

<rtc-room> can also be implemented in Windows, iOS, and Android platforms. Available references are listed in the following table. For more information about how it works, please see Design Documentation.

Platform	SDK Download	API Documentation
Windows(C++)	DOWNLOAD	API
Windows(C#)	DOWNLOAD	API
IE browser	DOWNLOAD	API
iOS	DOWNLOAD	API
Android	DOWNLOAD	API

Recording

- Step 1: Activate Tencent Cloud VOD service.
- Step 2: Log in to the LVB console (based on which the Mini Program audio/video streaming media is built), enable recording feature by going to **Access Management** -> **Access Configuration** -> **LVB Recording**. (Note: The recording fee is charged by the number of concurrent LVB recordings.)

- step3: These recorded files can be found on the Video Management interface of VOD, or obtained by calling the REST API of VOD.

live-room

Last updated：2018-07-11 11:40:53

Tag Overview

The **<live-room>** tag implemented based on <live-pusher> and <live-player> is a custom component used for audio/video chats among two or more people. It is mainly suitable for one-to-many audio/video chat scenarios (unlike <rtc-room>).

Demo

You can scan the following QR code or search **Tencent Video Cloud** in WeChat Mini Program to open the demo mini program. The **Live Room** in the demo is the typical application scenario of <live-room>.

Tag Description

Attributes

Attribute	Type	Value	Description
template	String	'1v1' or '1v3'	(Required) Identifies the UI template used by components (If you need to customize the UI, please see UI Customization)
roomID	String	Custom	(Optional) Room ID (roomID cannot be empty when role = audience)
roomName	String	Custom	(Optional) Room name
role	String	'presenter' or 'audience'	(Required) "presenter" represents VJ and "audience" represents audience
pureaudio	Boolean	true or false	(Optional) Whether to enable audio-only push. Default is false.
beauty	Number	0-5	(Optional) Beauty filter level: 0-5. Default is 5
muted	Boolean	true or false	(Optional) Whether to mute. Default is false
debug	Boolean	true or false	(Optional) Whether to print push debug information. Default is false.
bindonRoomEvent	function		(Required) Listens the events returned by RTCRoom

APIs

The **<live-room>** component contains the following APIs. You need to obtain the reference of the <live-room> tag using selectComponent before performing appropriate operations.

Function Name	Description
start()	Start
pause()	Pause
resume()	Resume
stop()	Stop
requestJoinPusher()	Request for joint broadcasting; applicable to audience
respondJoinReq(agree:Boolean, audience:Object)	Allow the request for joint broadcasting; applicable to presenter

Function Name	Description
switchCamera()	Switch camera
sendTextMsg(text:String)	Sends text messages

```
var liveroom = this.selectComponent("#liveroomid")
liveroom.pause();
```

Event notification

The **<live-room>** tag returns internal events through **onRoomEvent**. The format of the event parameters is as follows:

```
"detail": {
"tag": "The unique event tag ID",
"code": "Event code",
"detail": "Detailed parameters of the event"
}
```

Sample Code

```
//Page.wxml file
<live-room id="liveroomid"
roomID="{{roomID}}"
roomName="{{roomName}}"
template="1v3"
beauty="{{beauty}}"
muted="{{muted}}"
debug="{{debug}}"
bindonRoomEvent="onRoomEvent">
</rtcroom>


//Page.js file
Page({
data: {
//...
roomID: '',
roomName: '',
beauty: 3,
muted: false,
debug: false,
```

```
},
//...
onRoomEvent: function(e){
switch(e.detail.tag){
case 'roomClosed': {
//Room is closed
break;
}
case 'error': {
//An error occurred
var code = e.detail.code;
var detail = e.detail.detail;
break;
}
case 'recvTextMsg': {
//Receive a text message
var text = e.detail.detail;
break;
}
case 'joinPusher': {
//Receive a request for joint broadcasting from audience
var audience = e.detail;
var name = audience.userName;
var id = audience.userID;
//Allow request
liveroom.respondJoinReq(true, audience)
break;
}
}
},

onShow: function () {
},

onHide: function () {
},

onRead: function() {
var liveroom = this.selectComponent("#liveroomid");
this.setData({
roomName: 'Test',
});
liveroom.start();
},

})
```

Usage Guide

Step 1: Activate appropriate cloud service

Audio/video capability for mini programs relies on Tencent Cloud LVB and IM services, which can be activated free of charge by clicking the link. IM service can be used immediately once activated. LVB service that has a high risk of posting pornographic and political content requires users to go through Tencent Cloud's manual audit.

Step 2: Download custom component source code

<live-room> is not a native tag provided by WeChat Mini Program, but a custom component. Therefore, you need additional code to support this tag. Click Mini Program Source Code to download the source code package, and you can obtain the required files under the `wxlite` folder.

Step 3: Log in to RoomService (Required)

Before using the **<live-room>** tag, log in first by calling `/utils/liveroom.js` to connect to RoomService in the backend.

```
var liveroom = require('/utils/liveroom.js');
...
liveroom.login({
serverDomain: '',
userID: '',
userSig: '',
sdkAppID: '',
accType: '',
userName: '' //Custom user nickname
});
```

For more information about how to enter parameters, please see DOC.

Step 4: Get room lists (optional)

If you want to use the existing room list of RoomService, rather than implementing a new one, you can obtain the list information by calling the `getRoomList` function of `/utils/liveroom.js`.

```
var liveroom = require('/utils/liveroom.js');
...
liveroom.getRoomList({
data: {
index: 0, //List index number
```

```
cnt: 20 //Number of lists to be pulled
},
success: function (ret) {
console.log('Get room lists:', ret.rooms);
},
fail: function (ret) {
console.error('Failed to get room lists :', ret);
}
});
```

Step 5: Introduce the component into the project

- Reference the component in the JSON configuration file under the page directory. This is required because <live-room> is not a native tag.

```
"usingComponents": {
"live-room": "/pages/liveroom_component/liveroom"
}
```

Step 6: VJ: Create a room

- Use the <live-room> tag in the wxml file under the page directory and specify the role as **presenter**.

```
<live-room id="liveroomid"
template="1v3"
role="presenter"
roomID="{{roomID}}"
roomName="Test room"
pureaudio="false",
beauty="5",
debug="true" >
</live-room>
```

- If you want to use the roomID assigned by RoomService, specify a roomName for <live-room>.

```
//Create a room (a roomID assigned by RoomService)
this.setData({
roomName: 'Test'
});
var liveroom = this.selectComponent("#liveroomid");
liveroom.start();
```

- If you want to specify a roomID, set the roomID attribute before calling the start() function.

```
//Create a room (a roomID that you specify)
this.setData({
roomID: 12345
});
var liveroom = this.selectComponent("#liveroomid");
liveroom.start();
```

- In either case, a room is created only when the **start()** function is called.

Step 7: Audience: Join a room

- Use the <live-room> tag in the wxml file under the page directory and specify the role as **audience**.

```
<live-room id="liveroomid"
template="1v3"
role="audience"
roomID="{{roomID}}"
pureaudio="false",
beauty="5",
debug="true" >
</live-room>
```

- If a room with a specified roomID has already been created, calling start() will enter the created room, rather than creating a new one.

```
//Create a room (a roomID that you specify)
this.setData({
roomID: 12345
});
var liveroom = this.selectComponent("#liveroomid");
liveroom.start();
```

Step 8: Initiate joint broadcasting

- Audience: initiate a request for joint broadcasting to VJs

```
var liveroom = this.selectComponent("#liveroomid");
liveroom.requestJoinPusher();
```

- VJ: Allow or reject the request for joint broadcasting

```
var liveroom = this.selectComponent("#liveroomid");
liveroom.respondJoinReq(true, aduience);
```

UI customization

If the default "1v1" and "1v3" screen layouts do not suite your needs, you can customize the screen based on the specific project:

- **Create a UI template**

```
//Step 1: Create a /pages/templates/mytemplate folder and mytemplate.wxml and mytemplate.wxss fil
es.

//Step 2: Write the mytemplate.wxml and mytemplate.wxss files.
//mytemplate.wxml
<template name='mytemplate'>
<view class='inner-container'>
<live-pusher wx:if="{{isCaster&&mainPusherInfo.url}}" id="pusher" mode="RTC" enable-camera="{{tru
e}}" url="{{mainPusherInfo.url}}" beauty="{{beauty}}" muted="{{muted}}" aspect="{{mainPusherInfo.as
pect}}" waiting-image="https://mc.qcloudimg.com/static/img/daeed8616ac5df256c0591c22a65c4d3/p
ause_publish.jpg"
background-mute="{{true}}" debug="{{debug}}" bindstatechange="onMainPush" binderror="onMain
Error">
<!-- <cover-image class='character' src="/pages/Resources/mask.png"></cover-image> -->
<cover-view class='character' style='padding: 0 5px;'>Me({{userName}})</cover-view>
</live-pusher>

<block wx:for="{{visualPlayers}}" wx:key="{{index}}">
<live-player wx:if="{{item.url}}" autoplay id="player" mode="{{item.mode}}" object-fit="fillCrop" src="
{{item.url}}" debug="{{item.debug}}" background-mute="{{item.mute}}" bindstatechange="onMainPla
yState" binderror="onMainPlayError">
<cover-view class='loading' wx:if="{{item.loading}}">
<cover-image src="/pages/Resources/loading_image0.png"></cover-image>
</cover-view>
<!-- <cover-image class='character' src="/pages/Resources/mask.png"></cover-image> -->
<cover-view class='character' style='padding: 0 5px;'>{{item.userName}}</cover-view>
</live-player>
</block>
</view>
```

```
<view class='list-container'>
<view class='.list-item-box' wx:if="{{!isCaster && linkPusherInfo.url}}">
<live-pusher wx:if="{{!isCaster && linkPusherInfo.url}}" id="audience_pusher" mode="RTC" url="{{link
PusherInfo.url}}" beauty="{{beauty}}" muted="{{muted}}"
aspect="{{linkPusherInfo.aspect ? linkPusherInfo.aspect : '3:4'}}" waiting-image="https://mc.qcloudim
g.com/static/img/daeed8616ac5df256c0591c22a65c4d3/pause_publish.jpg"
background-mute="true" debug="{{debug}}" bindstatechange="onLinkPush" binderror="onLinkErro
r">
<cover-image class='character' src="/pages/Resources/mask.png"></cover-image>
<cover-view class='character' style='padding: 0 5px;'>Me({{userName}})</cover-view>
<cover-view class='close-ico' bindtap="quitLink">x</cover-view>
</live-pusher>
</view>

<view class='.list-item-box' wx:for="{{members}}" wx:key="{{item.userID}}">
<live-player id="{{item.userID}}" autoplay mode="RTC" object-fit="fillCrop" min-cache="0.1" max-cac
he="0.3" src="{{item.accelerateURL}}" debug="{{debug}}" background-mute="{{true}}">
<cover-view class="close-ico" wx:if="{{item.userID == userID || isCaster}}" bindtap="kickoutSubPusher"
data-userid="{{item.userID}}">x</cover-view>
<cover-view class='loading' wx:if="{{false}}">
<cover-image src="/pages/Resources/loading_image0.png"></cover-image>
</cover-view>
<cover-image class='character' src="/pages/Resources/mask.png"></cover-image>
<cover-view class='character' style='padding: 0 5px;'>{{item.userName}}</cover-view>
</live-player>
</view>
</view>
</template>

//mytemplate.wxss
.videoview {
background-repeat:no-repeat;
background-size: cover;
width: 100%;
height: 100%;
}
```

- **Introduce a template into the live-room component**

```
//Add a custom template for the liveroom.wxml file in the <live-room> component
<import src='/pages/templates/mytemplate/mytemplate.wxml'/>
<view class='conponent-box'>
<view styles="width:100%;height=100%;" wx:if="{{template=='1v3' || template=='1v1'}}">
<template is='mytemplate' data="{{pushURL, aspect,
minBitrate, maxBitrate, beauty, muted, debug, members}}"/>
```

```
</view>
</view>

//Add a custom style for the liveroom.wxss file in the <live-room> component
@import "../templates/mytemplate/mytemplate.wxss";
```

Other platforms

《live-room》 can also be implemented in Windows, iOS, and Android platforms. Available references are listed in the following table. For more information about how it works, please see Design Documentation.

Platform	SDK Download	API Documentation
Windows(C++)	DOWNLOAD	API
Windows(C#)	DOWNLOAD	API
IE browser	DOWNLOAD	API
iOS	DOWNLOAD	API
Android	DOWNLOAD	API

Recording

- Step 1: Activate Tencent Cloud VOD service.
- Step 2: Log in to the LVB console (based on which the Mini Program audio/video streaming media is built), enable recording feature by going to **Access Management** -> **Access Configuration** -> **LVB**

Recording. (Note: The recording fee is charged by the number of concurrent LVB recordings.)

- Step 3: These recorded files can be found on the Video Management interface of VOD, or obtained by calling the REST API of VOD.

webrtc-room

Last updated：2018-10-09 16:30:35

Tag Overview

<webrtc-room> tag implemented based on <live-pusher> and ;live-player> is a custom component used for interconnection with WebRTC. If you want to use <live-pusher> and <live-player> for integration, or to understand how the <webrtc-room> works, please see DOC..

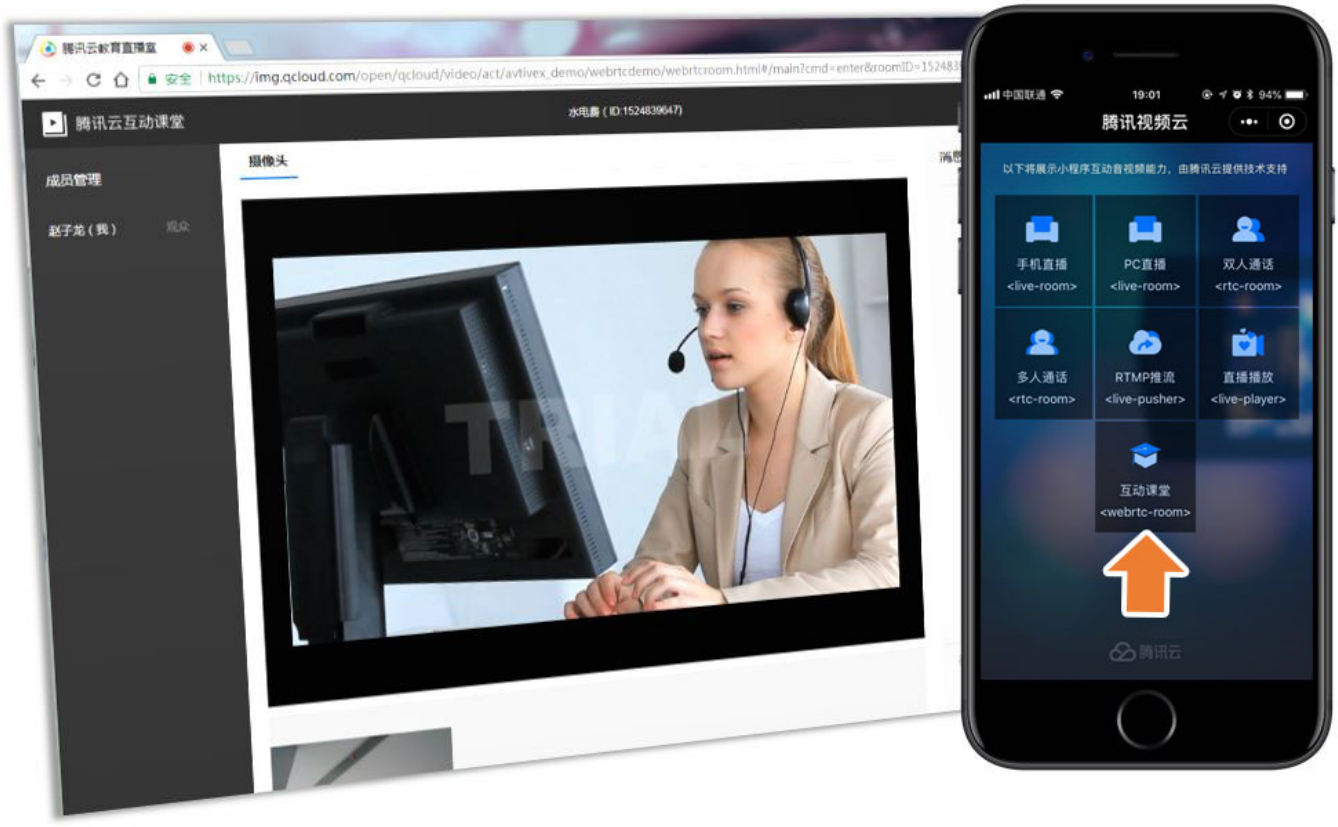
Version Requirement

- Support for WeChat 6.6.6.

Demo

- **PC**
  Open the experience page with the Chrome browser to use WebRTC for PC.

- **WeChat**
  Go to **Discover** -> **Mini Programs**, then search for **Tencent Video Cloud**, and click WebRTC feature tab to experience the intercommunication with Chrome for PC.



Source Code for Integration

Source Code for Integration	Description	Github Address
Mini Program source code	Includes component source code of <webrtc-room> and demo source code	Go
PC-end source code	Source code for the integration of WebRTC for Chrome implemented based on WebrtcAPI (where, component/WebRTCRoom.js implements a simple room management feature, and component/mainwindow.js contains the code for using WebRTC API)	Go
Backend source code	Implements a simple room list feature, and also contains the code for generating parameters required for <webrtc-room>	Go

Tag Description

Attributes

Attribute	Type	Value	Description
template	String	'1v3'	(Required) Identifies the UI template used by components (If you need to customize the UI, please see UI Customization)
sdkAppID	String	''	(Required) The AppID obtained when you activate the IM service
userID	String	''	(Required) User ID
userSig	String	''	(Required) Identity signature, which functions as a login password
roomID	Number	''	(Required) Room ID
privateMapKey	String	''	(Required) Room permission key, equivalent to the key for joining a room with the specified room ID
beauty	Number	0-5	(Optional) Beauty filter level: 0-5. Default is 5
muted	Boolean	true, false	(Optional) Whether to mute. Default is false
debug	Boolean	true, false	(Optional) Whether to print push debug information. Default is false
bindRoomEvent	function		(Required) Listens the events returned by <webrtc-room> component
enableIM	Boolean	true, false	(Optional) Default is false
bindIMEvent	function		Required when IM is enabled. It listens the events returned by IM

APIs

<webrtc-room> component contains the following APIs. You need to obtain the reference of the <webrtc-room> tag using selectComponent before performing appropriate operations.

Function Name	Description
start()	Start

Function Name	Description
pause()	Pause
resume()	Resume
stop()	Stop
switchCamera()	Switch camera

```
var webrtcroom = this.selectComponent("#webrtcroomid")
webrtcroom.pause();
```

Event notification

<webrtc-room> tag returns internal events through **onRoomEvent** and IM message events through **onIMEvent**. The format of the event parameters is as follows:

```
"detail": {
"tag": "The unique event tag ID",
"code": "Event code",
"detail": "Detailed parameters of the event"
}
```

Sample Code

```
//Page.wxml file
<webrtc-room id="webrtcroom"
roomID="{{roomID}}"
userID="{{userID}}"
userSig="{{userSig}}"
sdkAppID="{{sdkAppID}}"
privateMapKey="{{privateMapKey}}"
template="1v3"
beauty="{{beauty}}"
muted="{{muted}}"
debug="{{debug}}"
bindRoomEvent="onRoomEvent"
enableIM="{{enableIM}}"
bindIMEvent="onIMEvent">
</webrtc-room>
```

```
//Page.js file
Page({
data: {
//...
roomID: '',
userID: '',
userSig: '',
sdkAppID: '',
beauty: 3,
muted: false,
debug: false,
enableIM: false
},
onRoomEvent: function(e){
switch(e.detail.tag){
case 'error': {
//An error occurred
var code = e.detail.code;
var detail = e.detail.detail;
break;
}
}
},
onIMEvent: function(e){
switch(e.detail.tag){
case 'big_group_msg_notify':
//Receive a group message
console.debug( e.detail.detail )
break;
case 'login_event':
//Notification of login event
console.debug( e.detail.detail )
break;
case 'connection_event':
//Connection event
console.debug( e.detail.detail )
break;
case 'join_group_event':
//Notification of joining a group
console.debug( e.detail.detail )
break;
}
},

onLoad: function (options) {
self.setData({
```

```
userID: self.data.userID,
userSig: self.data.userSig,
sdkAppID: self.data.sdkAppID,
roomID: self.data.roomID,
privateMapKey: res.data.privateMapKey
}, function() {
var webrtcroomCom = this.selectComponent('#webrtcroom');
if (webrtcroomCom) {
webrtcroomCom.start();
}
})
},

})
```

Usage Guide

Step 1: Activate appropriate cloud service

The interconnection between Mini Programs and WebRTC is implemented based on the Tencent-RTC (TRTC)) service which needs to be activated.

- Log in to the TRTC console. If you have not activated this service, click **Apply** to go to Tencent Cloud's manual audit stage. The service is activated upon the approval of your application.

- After the service is activated, log in to the console, then create a TRTC application, and click **OK**.

- Get `sdkAppID, accountType, privateKey` used in Step 4 from the TRTC console:

Step 2: Download custom component source code

<webrtc-room> is not a native tag provided by WeChat Mini Program, but a custom component. Therefore, you need additional code to support this tag. Click the link to Mini Program source code to download the source code package, and you can obtain the required files under the `wxlite` folder.

Step 3: Introduce the component into the project

- Reference the component in the json configuration file under the page directory. This is required because <webrtc-room> is not a native tag.

```
"usingComponents": {
"webrtc-room": "/pages/webrtc-room/webrtc-room"
}
```

- Use the tag in the wxml file under the page directory.

```
<webrtc-room id="webrtcroomid"
roomID="{{roomID}}"
userID="{{userID}}"
userSig="{{userSig}}"
sdkAppID="{{sdkAppID}}"
privateMapKey="{{privateMapKey}}"
template="1v3"
beauty="{{beauty}}"
muted="{{muted}}"
debug="{{debug}}"
bindRoomEvent="onRoomEvent"
enableIM="{{enableIM}}"
bindIMEvent="onIMEvent">
</webrtc-room>
```

Step 4: Get key information

Get the key information according to the following table. These keys are required to use Tencent Cloud ILVB service.

KEY	Example	Role	How to Get
sdkAppID	1400087915	Used for billing and business differentiation	You can get it from Step 1
userID	xiaoming	User ID	It can be specified by your server or you can use the openid of Mini Program
userSig	Encrypted string	Equivalent to the login password corresponding to the userid	Issued by your server (PHP/JAVA)

KEY	Example	Role	How to Get
roomID	12345	Room ID	Specified by your server
privateMapKey	Encrypted string	Ticket for joining a room, equivalent to the key used to enter a room with the specified roomid	Issued by your server (PHP / JAVA)

Download sign_src.zip to get the calculation code for the server to issue userSig and privateMapKey. The algorithm for generating userSig and privateMapKey signatures is **ECDSA-SHA256**.

Step 5: Join the room

```
self.setData({
userID: userID,
userSig: userSig,
sdkAppID: sdkAppID,
roomID: roomID,
privateMapKey: privateMapKey
}, function() {
var webrtcroomCom = this.selectComponent('#webrtcroomid');
if (webrtcroomCom) {
webrtcroomCom.start();
}
})
```

UI customization

- **Create a UI template**

```
//Step 1: Create a /pages/templates/mytemplate folder and mytemplate.wxml and mytemplate.wxss fil
es.

//Step 2: Write the mytemplate.wxml and mytemplate.wxss files.
//mytemplate.wxml
<template name='mytemplate'>
<view class='videoview'>
<view class="pusher-box">
<live-pusher
id="rtcpusher"
autopush
```

```
mode="RTC"
url="{{pushURL}}"
aspect="{{aspect}}"
min-bitrate="{{minBitrate}}"
max-bitrate="{{maxBitrate}}"
audio-quality="high"
beauty="{{beauty}}"
muted="{{muted}}"
waiting-image="https://mc.qcloudimg.com/static/img/
daeed8616ac5df256c0591c22a65c4d3/pause_publish.jpg"
background-mute="{{true}}"
debug="{{debug}}"
bindstatechange="onPush"
binderror="onError">
<cover-image class='character' src="/pages/Resources/mask.png"></cover-image>
<cover-view class='character' style='padding: 0 5px;'>Me</cover-view>
</live-pusher>
</view>
<view class="player-box" wx:for="{{members}}" wx:key="userID">
<view class='poster'>
<cover-image class='set'
src="https://miniprogram-1252463788.file.myqcloud.com/roomset_{{index + 2}}.png">
</cover-image>
</view>
<live-player
id="{{item.userID}}"
autoplay
mode="RTC"
wx:if="{{item.accelerateURL}}"
object-fit="fillCrop"
min-cache="0.1"
max-cache="0.3"
src="{{item.accelerateURL}}"
debug="{{debug}}"
background-mute="{{true}}"
bindstatechange="onPlay">
<cover-view class='loading' wx:if="{{item.loading}}">
<cover-image src="/pages/Resources/loading_image0.png"></cover-image>
</cover-view>
<cover-image class='character' src="/pages/Resources/mask.png"></cover-image>
<cover-view class='character' style='padding: 0 5px;'>{{item.userName}}</cover-view>
</live-player>
</view>
</view>
</template>
```

```
//mytemplate.wxss
.videoview {
background-repeat:no-repeat;
background-size: cover;
width: 100%;
height: 100%;
}
```

- **Introduce a template into the webrtc-room component**

```
//Add a custom template for the webrtcroom.wxml file in the <webrtc-room> component
<import src='/pages/templates/mytemplate/mytemplate.wxml'/>
<view class='conponent-box'>
<view styles="width:100%;height=100%;" wx:if="{{template=='1v3'}}">
<template is='mytemplate' data="{{pushURL, aspect,
minBitrate, maxBitrate, beauty, muted, debug, members}}"/>
</view>
</view>

//Add a custom style for the webrtcroom.wxss file in the <webrtc-room> component
@import "../templates/mytemplate/mytemplate.wxss";
```

Integration with Chrome

Integration of H5 video chat in Chrome by referring to the WebrtcAPI , on the Tencent Cloud official website, which will not be discussed here.

Enterprise-based WebEXE

Last updated：2018-09-21 19:20:41

Solutions

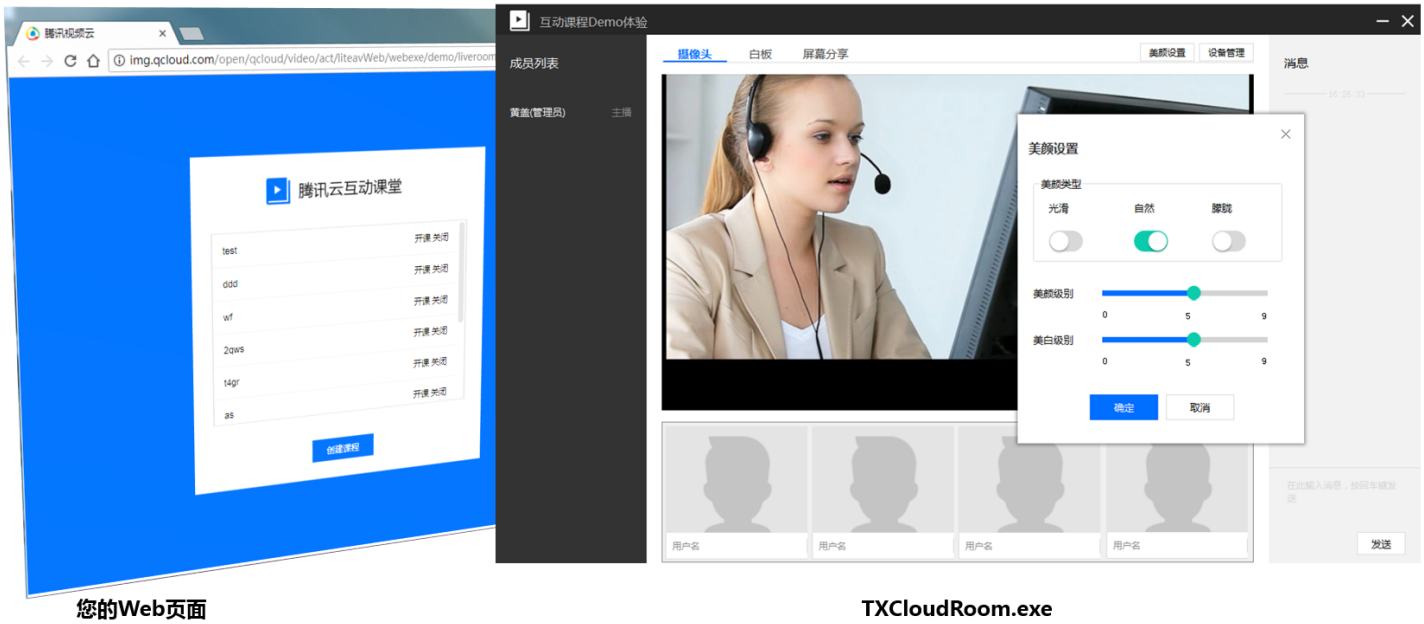
WebEXE and WebRTC are the two access solutions for enterprises. The following table lists the application scenarios and advantages/disadvantages of these two solutions. You can choose accordingly based on your own needs.

Solution	WebEXE	WebRTC
URL for documentation	DOC	DOC
Application scenario	To B (company employees)	To C (individual customers)
Advantage	Some advanced features can be implemented regardless of restrictions in browsers	It can be directly implemented in Chrome browser without the need to install plug-ins, which is suitable for ordinary users
Disadvantage	You need to install the program as prompted	Features are subject to the secure limits of Chrome browser
Beauty filter	Supported	Not supported
Desktop screencap	Supported	A screencap plug-in needs to be installed
Local recording	Supported	Not supported
Dependent cloud service	LVB + IM	TRTC + IM

Viewing Demo

Open the experience URL in a browser to get started with the WebEXE solution. The website on the left side can be replaced with your own Web page, and the TXCloudRoom.exe on the right side can be used to implement video chat and other features.

- **Web page (Web)**: Hosts the original business system and business logics, such as order system, CRM system, and other electronic stream systems.
- **Desktop application (EXE)**: Applications like WeChat for PC, which can be directly triggered by your Web page. Featuring high performance and good stability, it can implement features that cannot be implemented on some browsers.



Source Code Debugging

PC Web page

Click GitHub to download the source code for Web page. Double click index.html in a local browser to use and debug the appropriate features of WebEXE. When using it for the first time, you will be prompted to download and install local client plug-ins.

Directory	Description
index.html	Home page of demo
doubleroom.html	Demo page of one-on-one video chat
liveroom.html	Demo page of interactive video chat
assets	CSS style sheet and resource file used in demo page
js	The javascript used in demo page, where EXEStart.js is located.

Directory	Description
exe	TXCloudRoomSetup.exe installation package is included

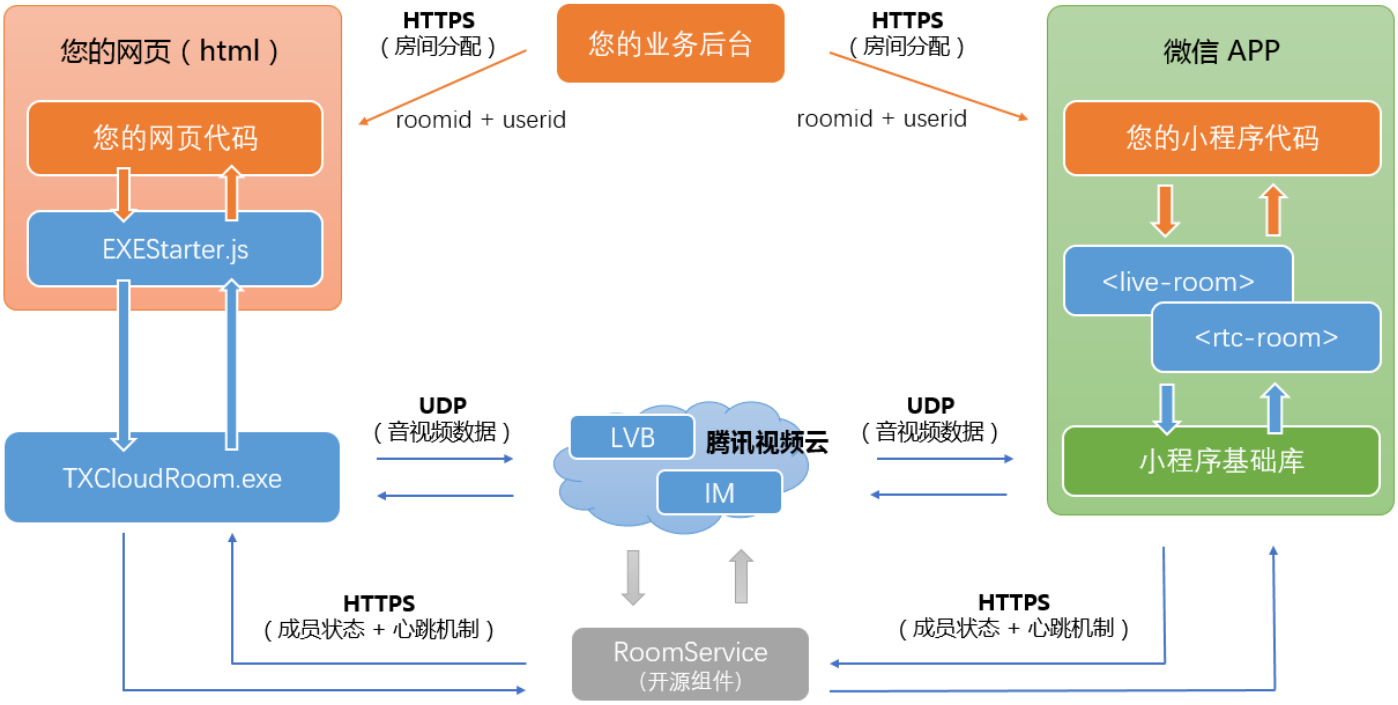
Server

Click GitHub to download server-end source code of **Java** version. This code is mainly used to implement a simple (unauthenticated) room list, to support features such as creating or closing a chat room. This code can be ignored if you just want to enable a video call (hard-code a roomid in PC Web page and Mini Program end respectively).

Directory	Description
README.pdf	Describes how to use this backend code
Backend API table.pdf	Provides a detailed description on how to implement this backend code internally
src	Backend room list source code of java version

Solution Integration

The following figure describes how to integrate WebEXE solution to your existing business system:



Step 1: Build a business server

The business server is mainly used to send roomid, userid, usersig and other information required to implement a video chat to PC Web pages and WeChat mini programs. The roomid and userid can be specified by your business backend. You must ensure that these IDs cannot overlap with each other. For more information on how to calculate the usersig, please see DOC. We also provide the calculation source code of java and php versions.

Step 2: Deploy RoomService

Both LiveRoom (LVB+Joint broadcasting) and RTCRoom (Video chat) components used by WebEXE to implement a video chat rely on a backend open source component (JAVA | Node.js) called RoomService to implement room logics. Click RoomService.zip to download the appropriate code. For more information on how to deploy the code, please see **README.pdf** in the zip package.

Step 3: Integrate PC Web-end code

Your Web page needs to include the javascript file **EXEStarter.js** , and sends the roomid, userid, usersig obtained in Step 1 to the **startEXE** function of **EXEStarter.js**. Some of the key parameters are described as follows:

Parameter	Description

Parameter	Description
sdkAppID	Tencent Cloud IM service uses sdkAppID to identify IM users. For more information on how to get this value, please see DOC .
accType	It was used to distinguish App types and is now saved for compatibility reasons. For more information on how to get this value, please see DOC.
userID	User ID, which is assigned by your business server. Users will be forced offline in case of overlapped IDs.
userSig	Equivalent to the user password. For more information on how to calculate this value, please see DOC .
serverDomain	RoomService address, which is obtained after the RoomService is deployed in Step 2.
roomId	Room ID, which is assigned by your business server. Users can start a video chat if they join a room with the same ID either from mini program end or PC end.
type	RTCRoom and LiveRoom modes. Differences can be found in Step 4.
template	Layout of video windows. Default is 1V1. For more information, please see Template .
mixRecord	Cloud recording. In this mode, EXE does not participate in recording tasks which will be executed at the backend. Therefore, both WebEXE and WebRTC are applicable. However, customization is not supported.
screenRecord	Local recording mode specific to WebEXE. EXE program directly captures screen images locally and generates a recorded video in real time. Depending on different parameters, EXE will store the generated video files locally or push them onto the cloud.
cloudRecordUrl	If you choose RecordScreenToServer or RecordScreenToBoth for screenRecord, you need to specify a rtmp:// push URL. In this case, the video stream can record the video content to the cloud according to the standard push and recording modes.

EXEStarter.js is mainly used to trigger the TXCloudRoom.exe desktop program and implement a two-way communication with TXCloudRoom.exe. Your Web page only needs to process room management logics, and TXCloudRoom.exe handles complicated audio/video features.

Click sample code, and you can find a program used to trigger TXCloudRoom.exe. You can also obtain well-designed source code applicable to PC Web page from GitHub.

Step 4: Integrate Mini Program-end code

For the integration of Mini Program-end code, please see documents about integration of WeChat-end code:

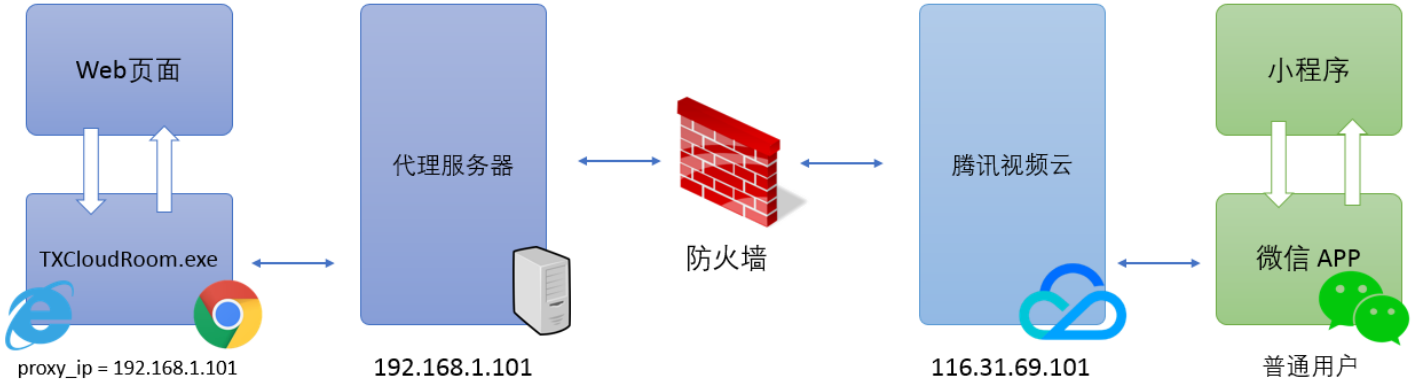
Document URL	Application Scenario
<rtc-room>	Scenarios of video chat only, such as one-on-one video chat, or video conference
<live-room>	Combined scenarios of LVB+Joint broadcasting implemented based on LVB service, to allow thousands of viewers to watch LVB concurrently at a very low bandwidth cost and support real-time chat among VJs and viewers.

How to Record LVB

You can record a user's entire LVB process as a video file for replay. For more information on how to implement the recording feature, please see Record Entire LVB..

NAT Traversal

Many enterprises have a secure gateway to block Internet access of enterprise internal network. But all Tencent Cloud solutions are accessed over the Internet. To solve this problem, a proxy server is required. For more information on how to solve this problem, please see DOC .



Record

Last updated：2018-07-11 11:51:45

Feature Description

You can record a user's entire LVB process as a video file for replay. It is applicable to online loss assessment, interactive classroom, remote court hearing, and other scenarios.

Enabling Recording

Recording & Replay is built on Tencent Cloud's **VOD service**. To use this feature, you need to Activate VOD Service on Tencent Cloud console. After the service is enabled, you can find the new recorded files in Video

Management on VOD console.

How to enable recording? Two methods are available:

1. Enable recording globally

You can enable or disable recording for all LVB streams in **LVB Console -> Access Management -> LVB Code Access -> Access Configuration**, as shown below:

Notes:

- The video wrapper formats supported by the global recording feature are MP4, HLS and FLV.
- You can specify multiple formats in which the video is recorded at a time (mobile browsers only support playback of MP4 and HLS videos).
- Changing resolution or switching between landscape/portrait modes during LVB is not supported for MP4 videos. If you need the screen sharing feature (screen resolution may change during push) without stream mixing enabled, HLS format is recommended.
- If the specified recording format is FLV or MP4, the length of a video fragment for global recording defaults to 30 minutes. Submit a ticket if you need to configure the length of a video fragment for global recording.
- HLS (m3u8) file is on a fragmentation basis in essence, so you can always get a single m3u8 file as long as no push interruption occurs during LVB. But in case of a push interruption during LVB, fragmentation will occur in the process of recording, which means you will get multiple m3u8 files. One of the common problems is switching of App to background. To solve this problem, you're recommended to allow push at the background.

2. Specify a room for recording

If global recording is not enabled, you can still enable recording for some important video streams by setting the record parameter in custom to true when calling the API createExeAsRoom in the EXEStarter.js of Web end.

Sample code:

```
EXEStarter.createExeAsRoom({
userdata: {
userID: accountInfo.userID,
userName: accountInfo.userName,
sdkAppID: accountInfo.sdkAppID,
accType: accountInfo.accountType,
userSig: accountInfo.userSig,
},
roomdata: {
serverDomain: "https://room.qcloud.com/",
roomAction: object.data.roomAction,
roomID: object.data.roomID,
roomName: object.data.roomName,
roomTitle: object.data.roomTitle,
roomLogo: "http://liteav.myqcloud.com/windows/logo/liveroom_logo.png",
type: EXEStarter.StyleType.LiveRoom,
template: EXEStarter.Template.Template1V3,
},
custom: { //Optional
record: true, //Record the current video stream at the background
},
success: function (ret) {
console.log('EXEStarter.openExeRoomLocal application launched successfully');
},
fail: function (ret) {
console.log('EXEStarter.openExeRoom Failed to launch local application');
if (ret.errCode == -1) {
//Local application is not installed
}
},
})
```

Notes:

- The video wrapper format used by the recording feature is MP4 by default (convenient for playback, download and transmission).
- The length of a video fragment of an MP4 file is 2 hours by default. The LVB content within 2 hours will be recorded into an MP4 file as long as no push interruption occurs.

You can choose either one of these two methods based on your business needs. If you need to record every video stream, global recording is recommended. If you want to record mixing images into a file, you are recommended to specify a room for recording.

Getting Files

When a new recorded video file is generated, a playback URL will be generated. You can implement many extensions based on your business scenarios. For example, you can add the URL to historical information for archiving, or put it in the replay list to recommend a high-quality video to your App users after manual filtering.

You can get the URL of the recorded file by the following three ways:

1. Listen to notification passively

You can use Tencent Cloud **Event Notification Service**: Register a **callback URL** for your server on Tencent Cloud which will notify you of the generation of a new recorded file using this URL.

The following is a typical notification message, which indicates: a new FLV recorded file with ID 9192487266581821586 has been generated, and the playback URL is: `http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9e61c2624af7.f0.flv` .

```
{
"channel_id": "2121_15919131751",
"end_time": 1473125627,
"event_type": 100,
"file_format": "flv",
"file_id": "9192487266581821586",
"file_size": 9749353,
```

```
"sign": "fef79a097458ed80b5f5574cbc13e1fd",
"start_time": 1473135647,
"stream_id": "2121_15919131751",
"t": 1473126233,
"video_id": "200025724_ac92b781a22c4a3e937c9e61c2624af7",
"video_url": "http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9e61c2624af7.f0.flv"
}
```


Note: The time information sent from the App client is important. If you want to define that all the files recorded during this time period belong to an LVB, you just need to retrieve the recording notifications you received using LVB code and time information (each recording notification event comes with stream_id, start_time and end_time).

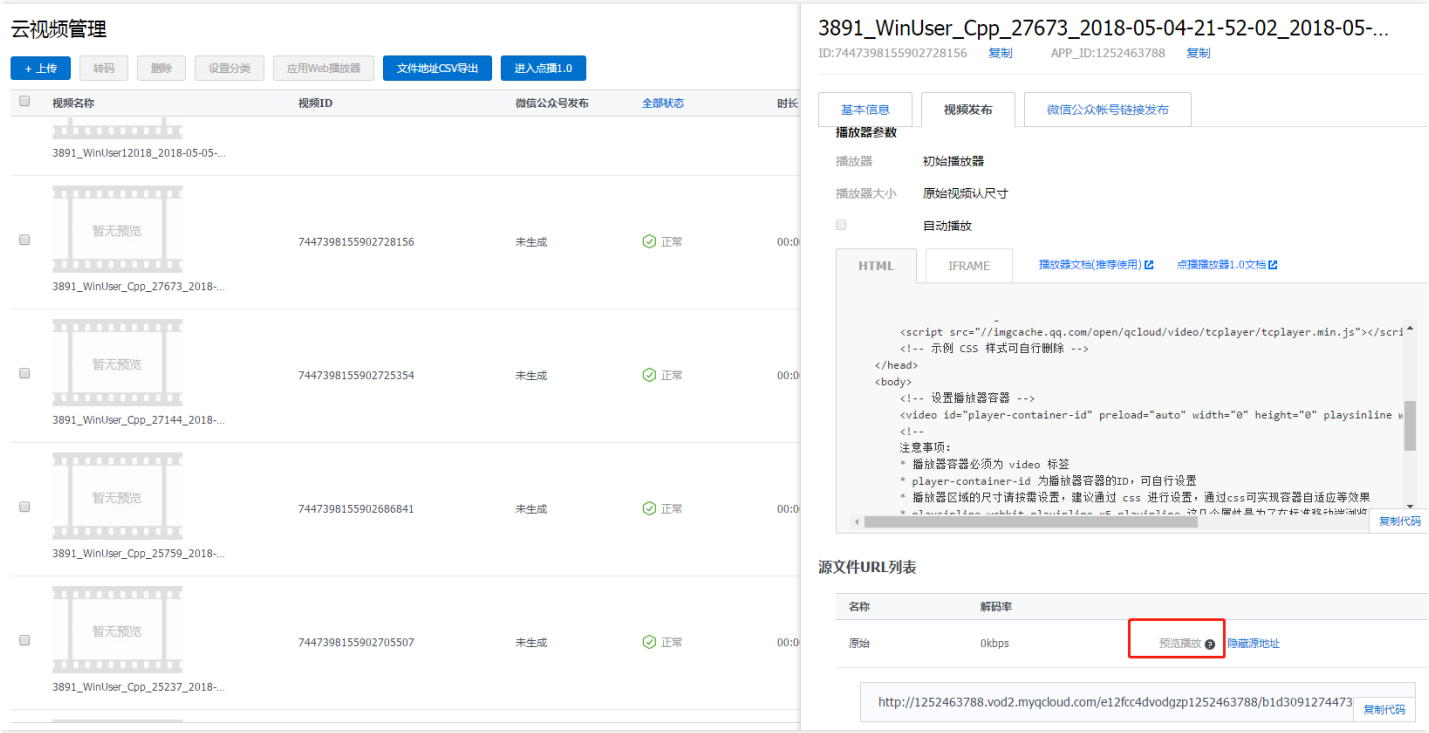
2. Query files actively

You can check if any new recorded file is generated on a regular basis using Tencent Cloud's query API (**Live_Tape_GetFilelist**). However, this method is not recommended for frequent use due to its unsatisfactory real-timeness and reliability since it has a slow response in case of a query for a large number of channels and cannot be called at a high frequency (only suitable for the channels that have just finished).

3. View files in video management

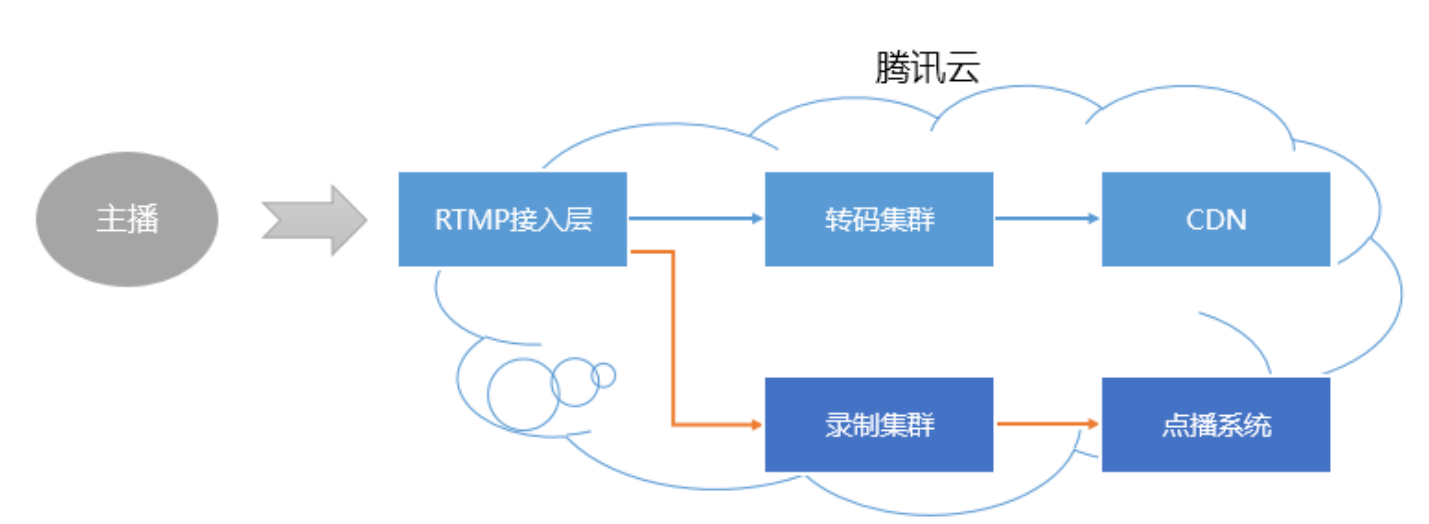
All recorded videos can be queried in Video Management on VOD console, where you can retrieve the content of video files. You can also search for specified videos by prefix, as shown below:

Select a video file in the video list, and you can find its details at the right side. Switch to the **Video Publishing** tab, and click **Display source address** to get the file URL. See the figure below:



FAQ

1. How does LVB recording work?



When you enable the recording for an LVB stream, the audio/video data is bypassed to the recording system. Every frame pushed from VJ's mobile phone is written into the recorded file by the recording system.

If an LVB push is interrupted, the access layer will immediately notify the recording server to record the file being written, store it into the VOD system and generate an index. Then you can find the new recorded file in the VOD system. If you have configured recording event notification on a server, the recording system will send the **index ID** and **online playback URL** to the server.

2. How many recorded files are generated in an LVB process?

- If the duration of an LVB is too short (for example, shorter than 1 second), no recorded file is generated.
- If the duration of an LVB is not long (shorter than 7 days or the length of an MP4/FLV fragment), and the push is not interrupted during LVB, only one recorded file is generated.
- If the duration of an LVB is very long (longer than 7 days (HLS format) or the length of an MP4/FLV fragment), the video is forcibly fragmented to avoid the time uncertainty of the flow of the file with a longer duration in a distributed system.
- If the push is interrupted during an LVB (SDK will re-push later), a new fragment is generated every time the interruption occurs.

3. How to stitch fragments?

Tencent Cloud supports stitching video fragments with cloud API. For more information on how to use this API, please see Video Stitching.

FAQ
COMMON PROBLEM

Last updated：2018-07-10 14:28:09

Why do <live-pusher> and <live-player> not work?

For policy and compliance considerations, <live-pusher> and <live-player> are not supported by all WeChat mini programs:

- Mini Programs of personal and enterprise accounts only support the categories in the following table:
- For Mini Programs meeting requirements of categories, you need to enable the component permissions in "Settings" -> "API Settings" of the Mini Program management backend.

Primary Category	Sub-category
"Social"	LVB
"Education"	Online education
"Healthcare"	Internet hospital and public hospital
"Government Affairs and Livelihood"	All secondary categories
"Finance"	Funds, trusts, insurance, banking, securities/futures, micro-credit of non-financial institutions, credit investigation, and consumer finance

Note: If your Mini Program still does not work after the settings are correctly made, that may be because the cache within the WeChat is not updated. Delete the mini program, restart WeChat, and try again.

How do I choose the right resolution ratio?

The resolution ratio is set by aspect. Available resolution ratios include 3:4 and 9:16, which are set according to the display area ratio of the current push and the playback view on the mobile phone. The mobile phone screen is usually in a resolution ratio of 9:16 when it is in portrait screen mode. If a person pushes through the full screen, 9:16 is set. If two persons are displayed side by side, one for push and the other for playback, and the display area is in a resolution ratio of 3:4, and then 3:4 is set.

How do I choose from SD, HD, FHD, and RTC modes for various applications, and how do I set the minimum and maximum bit rates?

Please see the <live-pusher> Tag Parameter Setting section for how to select the right mode and make bitrate settings.

How do I listen to key push and playback events?

See <live-pusher> Tag Internal Events section for how to listen to push events.
See the <live-player> Tag Internal Events section for how to listen to playback events.

The <live-push> tag cannot be inserted to certain mobile phones, is that true?

This problem has been identified and will be fixed in the next version. There is no way to capture this anomaly and prompt the user to allow access to the microphone and camera.
If you see a log message that says "insertLivePusher:fail:system permission denied", that means that no access to the microphone and camera is allowed. Check the microphone and camera permissions in the settings to ensure access to the microphone and camera is allowed.

Why does the Mini Program <live-play> tag occasionally turn black or fail to play?

You need to first understand the Page Lifecycle of WeChat Mini Programs by viewing Mini Program Page Life Cycle.

For the page lifecycle of mini programs, onLoad only loads data but not renders page, at which time the <live-push> and <liveplay> tags have not been created. Therefore, the method of obtaining or calling livepushercontext and liveplayercontext is uncertain. onReady indicates that the page has been loaded and rendered for the first time. Operations related to <live-pusher> and <live-player> must be implemented in onReady.
Examples:

```
onReady: function () {
var self = this;
this.data.videoContext = wx.createLivePlayerContext("video-livePlayer");
this.setData({
playUrl: "rtmp://live.hkstv.hk.lxdns.com/live/hks",
}, function () {
self.data.videoContext.stop();
self.data.videoContext.play();
})
},
```

How do I solve the log overlay problem when setting style="display:block" for tag <live-player>?

This is an identified problem. If you want to control the showing/hiding of tag <live-player> using the "display" attribute of "style", it is recommended that the hiding setting be style="display:none". For

showing, the default value is used for "display" without making any changes.

8. The same <live-player> tag is used to playback different URLs, but it does not work. How can I solve this?

When in autoplay mode, playUrl changes are not automatically played back. When not in autoplay mode, call the play function after changing playUrl. Therefore, no matter in which mode, when playUrl is changed to play another URL, you are recommended to do the following:

```
onPlay: function () {
var self = this;
this.data.videoContext = wx.createLivePlayerContext("video-livePlayer");
this.setData({
playUrl: "rtmp://live.hkstv.hk.lxdns.com/live/hks",
}, function () {
self.data.videoContext.stop();
self.data.videoContext.play();
})
},
```

Make sure that videoContext is called in setData's callback, and stop must be called before start.

The mode attribute is dynamically set for the <live-player> tag, but it does not work. How can I solve this?

The dynamically set mode does not become effective dynamically and you need to stop and start again. Please see the following for how to use it:

```
onChangeMode: function () {
var self = this;
this.data.videoContext = wx.createLivePlayerContext("video-livePlayer");
if (this.data.mode == "live") {
this.setData({ mode: "RTC"}, function () {
self.data.videoContext.stop();
self.data.videoContext.play();
});
}
else {
this.setData({ mode: "live"}, function () {
self.data.videoContext.stop();
self.data.videoContext.play();
});
}
},
```

Why does the muted attribute fail to function for some mobile phones when the <live-player> tag of Android Mini Program is in RTC mode?

This is an identified bug. We will fix it in the SDK as soon as possible and make it ready for the next release of WeChat.

The enable-camera attribute is dynamically set for the <live-pusher> tag, but it does not work. How can I solve this?

The enable-camera attribute does not become effective dynamically and needs to be set in advance. That is, dynamic enabling/disabling of the camera in the push process is not supported. This attribute must be set before push. To make it effective dynamically, you are recommended to do the following:

```
onEnableCamera: function () {
var self = this;
this.data.videoContext = wx.createLivePusherContext("video-livePusher");
this.setData({
enable-camera: "true"
},function () {
self.data.videoContext.stop();
self.data.videoContext.start();
});
},
```

Clean the <live-pusher> and <live-player> on the onUnload page.

Please see the following for details:

```
onUnload: function () {
self.data.pusherContext && self.data.pusherContext.stop();
self.data.playerContext && self.data.playerContext.stop();
},
```

When cover-view is overlaid on top of the <live-pusher> or <live-player> tag, it is not recommended to change the size of the tag.

When the size of <live-pusher> or <live-player> changes, iOS Mini Program cannot change the cover-view overlaid on it, and the interface display is unpredictable. Therefore, when cover-view is overlaid on the <live-pusher> or <live-player> tag, do not dynamically change the size of the tag.

What is <cameraview> in the Demo source code used for?

Because only the <live-pusher> and <live-player> tags are used to support real-time audio/video chats (especially multi-person video chats), substantial development effort is required. Therefore, based on the

native **custom component** capability of WeChat Mini Program, we have encapsulated complex status management and business logics in the custom tag called <cameraview>. You can learn about how to use <cameraview> and <live-player> to quickly build a two-person audio/video chat feature by viewing the codes in `doubleroom\room\room.js` from the source code package.