Real-time Communication

Client Features

Product Introduction

Contents

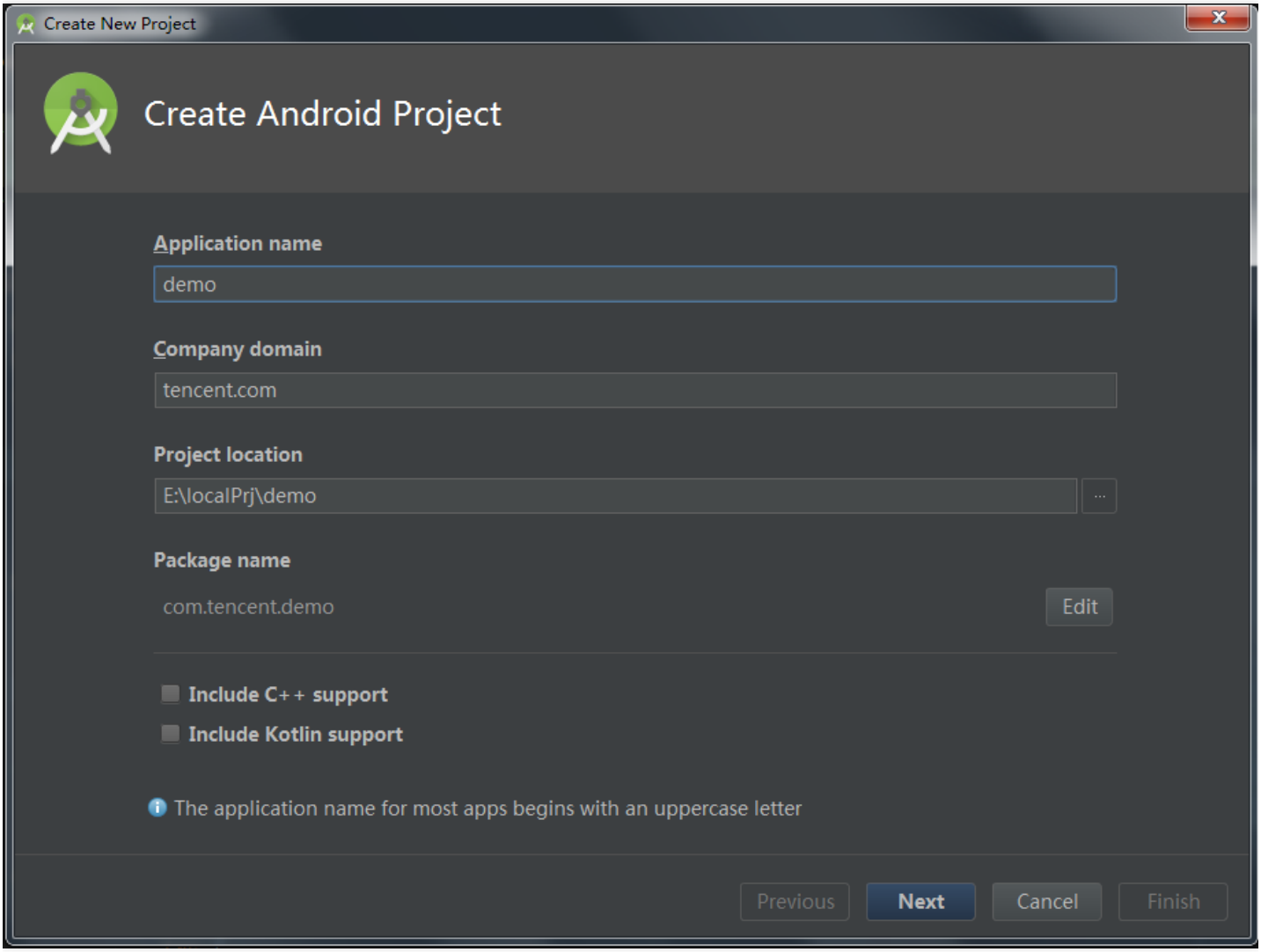
Client Features
Integrate SDK
Android

Last updated：2018-09-28 16:58:43

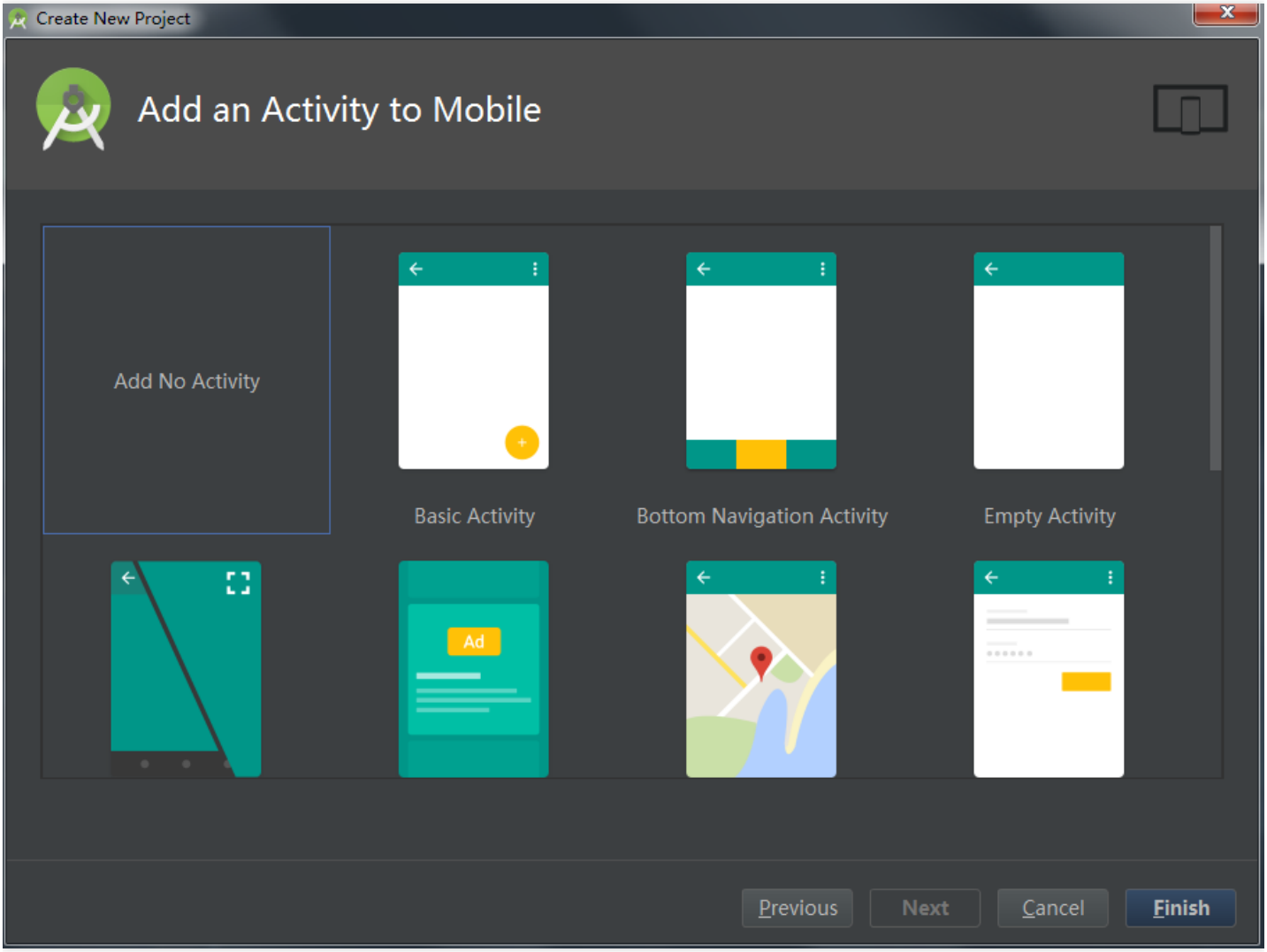
Procedure

Create an Android project

Open Android Studio, click the **File** menu, and select **New Project** to create a project:



Create an empty project:



Add a dependency (integrate SDK)

Modify the build.gradle file and add the dependency of iLiveSDK in "dependencies":

compile 'com.tencent.ilivesdk:ilivesdk:latest.release' //latest.release refers to the latest iLiveSDK version number

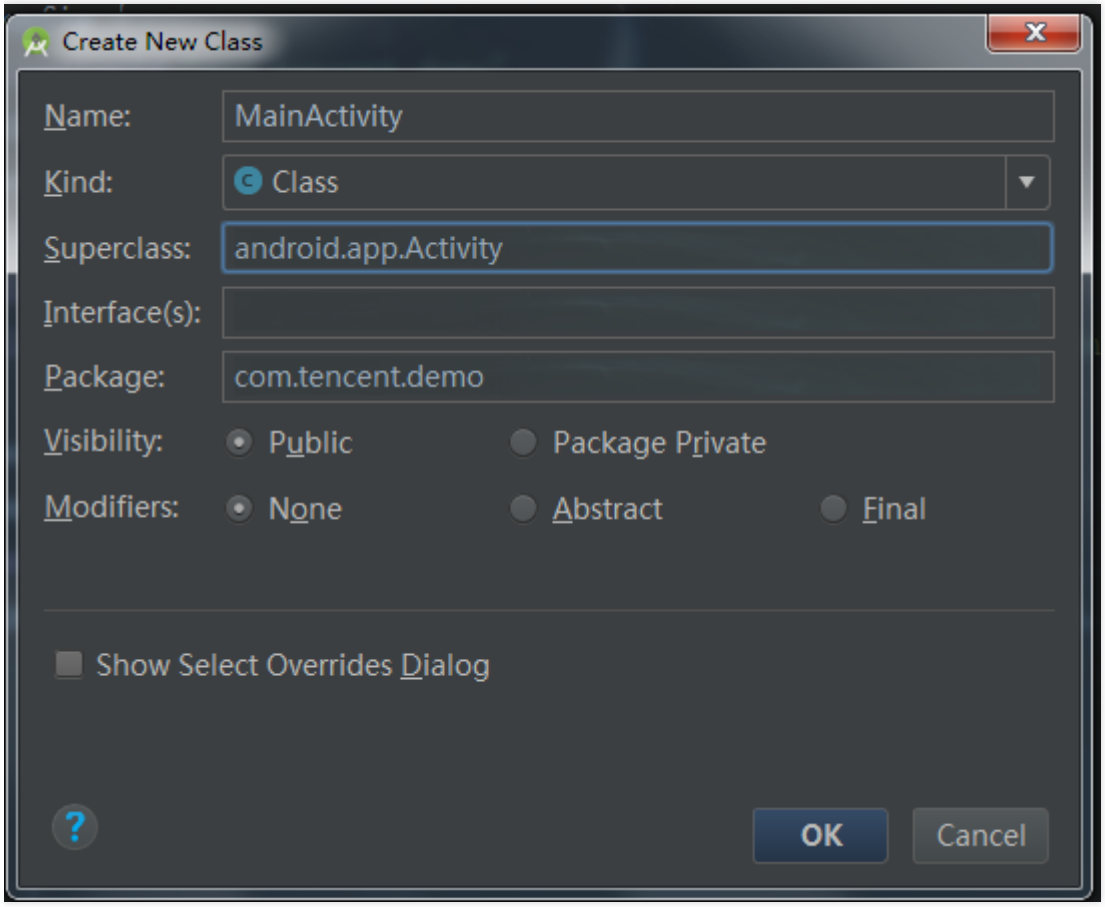
Create an application

Create a simple layout main.xml in "layout":

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">
```

```
<TextView
android:id="@+id/tv_version"
android:textColor="@color/colorAccent"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
</LinearLayout>
```

At the same time, create an Activity project:



Use the layout in application creation and output the version number of the iLiveSDK:

```
public class MainActivity extends Activity {
private TextView tvVersion;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
tvVersion = (TextView)findViewById(R.id.tv_version);

tvVersion.setText(" iLiveSDK: "+ILiveSDK.getInstance().getVersion()+"\n IMSDK:"+
TIMManager.getInstance().getVersion()+"\n AVSDK:"+
```

```
AVContext.sdkVersion);
}
}
```

Declare the application

Declare the application in AndroidManifest.xml:

```
<activity android:name=".MainActivity" android:screenOrientation="portrait" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
```

Compile and run the project

Compile and run the project, and then the following displays:

```
iLiveSDK:1.8.5
IMSDK:2.5.6.11073.11080
AVSDK:1.9.8.2
```

You have successfully integrated iLiveSDK.

FAQ

Failed to download aar

```
Error:Could not resolve all files for configuration ':app:debugCompileClasspath'.
> Could not resolve com.tencent.ilivesdk:ilivesdk:1.8.3.
Required by:
project :app
> Could not resolve com.tencent.ilivesdk4:ilivesdk:1.8.3.
> Could not get resource 'https://jcenter.bintray.com/com/tencent/ilivesdk/ilivesdk/1.8.4/ilivesdk-1.8.3.pom'.
> Could not GET 'https://jcenter.bintray.com/com/tencent/ilivesdk/ilivesdk/1.8.4/ilivesdk-1.8.3.pom'.
> Connect to jcenter.bintray.com:443 [jcenter.bintray.com/75.126.118.188] failed: Connection timed out: connect
Check the network first. Then, check whether the jcenter website is accessible by clicking on the above link. If a proxy is required, check if it is configured in gradle.properties.
```

Methods not found due to proguard

When calling some APIs, add the following configurations to enable the proguard for your project:

```
-keep class com.tencent.**{*;}
-dontwarn com.tencent.**

-keep class tencent.**{*;}
-dontwarn tencent.**

-keep class qalsdk.**{*;}
-dontwarn qalsdk.**
```

Crash due to multiple architectures

Only the armeabi architecture is supported (arm-v7a is supported for version 1.0.5 or later). If the project (or dependent library) has multiple architectures, the following configurations are required in build.gradle:

```
android{
defaultConfig{
ndk{
abiFilters 'armeabi', 'armeabi-v7a'
}
}
}
```

Email

If you have any questions, send us an email to trtcfb@qq.com.

iOS

Last updated：2018-10-09 10:03:01

This document describes how to integrate the TRTC SDK to an iOS device.

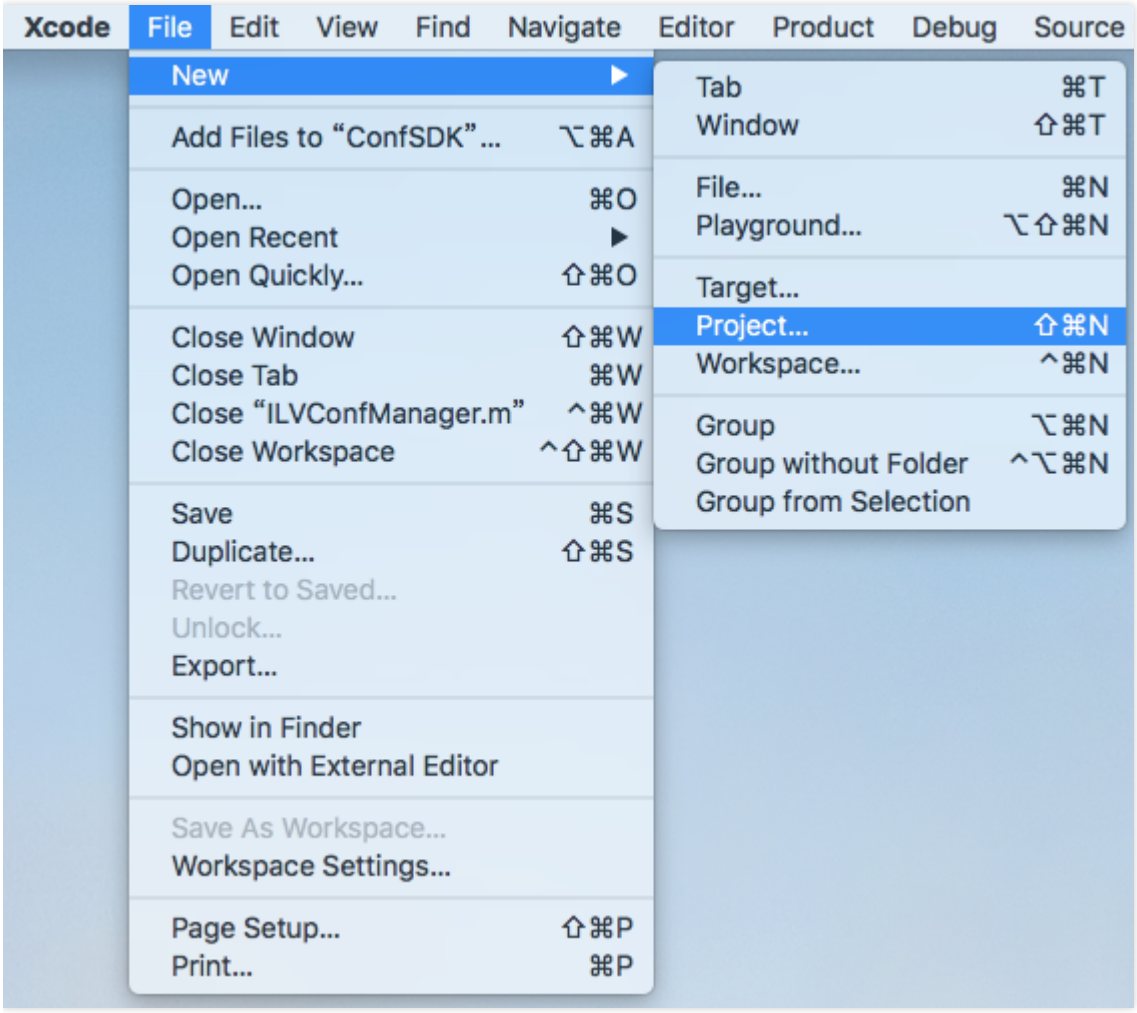
Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

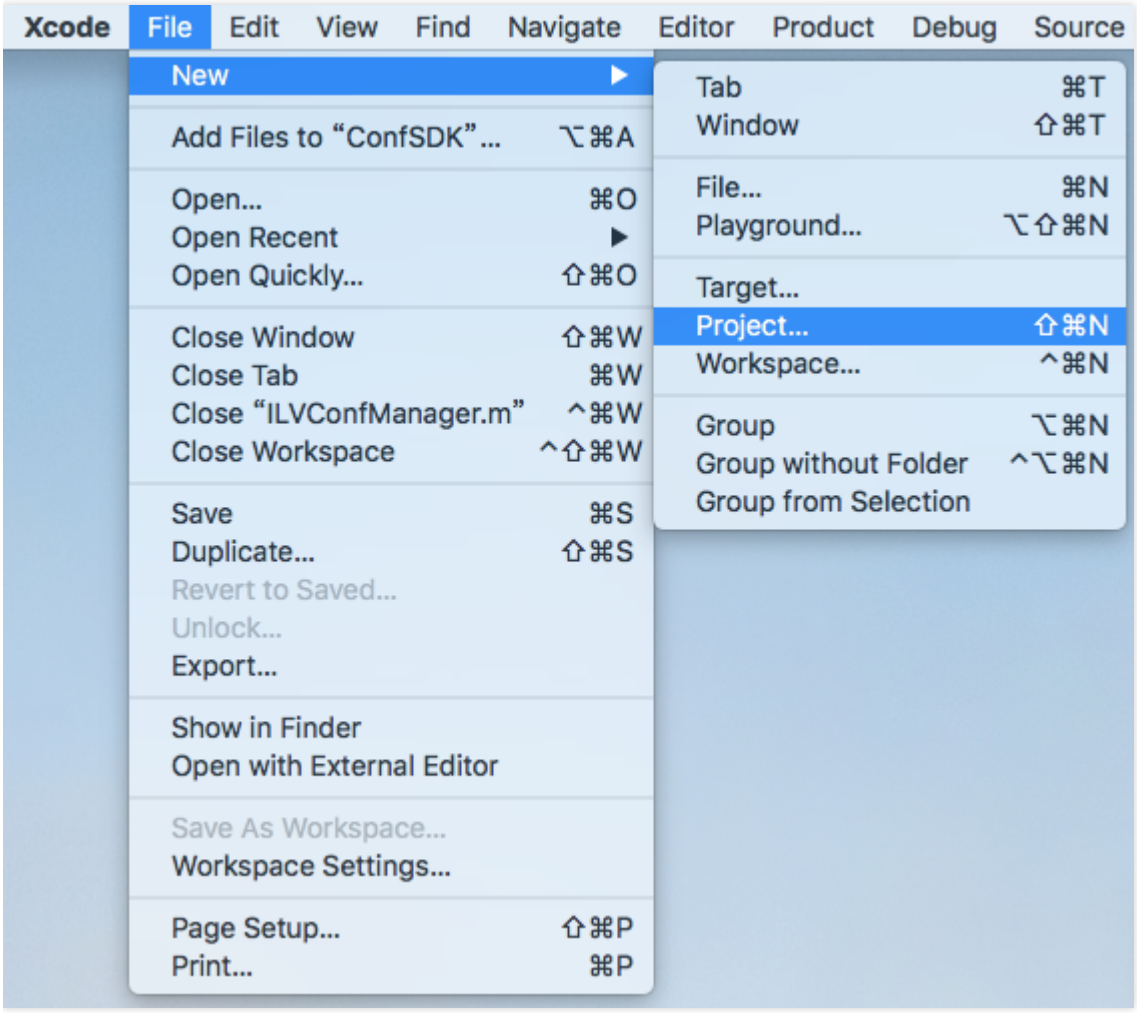
Procedure

Create an iOS project

If you already have a project to integrate, skip to Integrate the SDK.
First, create a project with Xcode to integrate the SDK.
Open Xcode and click **File** -> **New** -> **Project**:



Select **Single View App**



Name the project as Demo01_integration SDK and select Objective-C as Language. Enter Team, Organization Name and Organization Identifier as needed. Click **Next**. Select a location to store the project, and then click **Create**.

Integrate the SDK

Obtain the SDK

ILiveSDK is comprised of the following SDKs:

- BeautySDK: Provides beauty preprocessing
- IMSDK: Provides IM (Instant Messaging)
- AVSDK: Provides the underlying audio/video features
- ILiveSDK: Encapsulates audio/video APIs based on AVSDK to make them easy to use
- TILLiveSDK: Encapsulates LVB APIs based on ILiveSDK to allow users to access LVB features easily and quickly

  Create a folder named ILiveSDK in the project directory to store the SDK.

Import the SDK

After the download is completed, right-click on **Project** -> **Add Files to "Demo01_integration SDK"** to import the SDK to the project:

Select the new **ILiveSDK** folder in the pop-up select box, and then click **Add**:

The added project directory is as follows:

Add system dependent libraries

Some system libraries relied on by SDKs in ILiveSDK need to be added to the project.
Click **PROJECT** -> **TARGETS** -> **General**. At the bottom of Linked Frameworks and Libraries section, click **+**, enter the system library name, and click **Add**.

List of system libraries to be added:

- Accelerate.framework
- AssetsLibrary.framework
- AVFoundation.framework
- CoreGraphics.framework
- CoreMedia.framework
- CoreTelephony.framework
- CoreVideo.framework
- ImageIO.framework
- JavaScriptCore.framework
- OpenAL.framework
- OpenGLES.framework
- QuartzCore.framework
- SystemConfiguration.framework
- VideoToolbox.framework
- libbz2.tbd
- libc++.tbd
- libiconv.tbd
- libicucore.tbd
- libprotobuf.tbd
- libresolv.tbd
- libsqlite3.tbd
- libstdc++.6.tbd
- libstdc++.tbd
- libz.tbd

All the added system libraries are stored in the Frameworks folder. To add these system libraries to your project in an easy manner, you can download our demo code (Click to download) and directly drag the

system libraries in the Frameworks folder to the Linked Frameworks and Libraries section under your project.

Configure the project

To use the SDK properly, you also need to make the following configurations:
-Configure ObjC
Build Settings -> **Other Linker Flags** -> **-ObjC**:

Configure Bitcode
Build Settings -> **Enable Bitcode** -> **No**:

Run and check the SDK

After the above steps are completed, you can use ILiveSDK. Add the code to the viewDidLoad function of ViewController.m to obtain the version number:

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

//Obtain the version number
NSLog(@"ILiveSDK version:%@",[[ILiveSDK getInstance] getVersion]);
NSLog(@"AVSDK version:%@",[QAVContext getVersion]);
NSLog(@"IMSDK version:%@",[[TIMManager sharedInstance] GetVersion]);

//Print results
2018-03-27 15:22:37.187181+0800 Demo01_integration SDK[8182:16625633] ILiveSDK version:1.8.3.13017
2018-03-27 15:22:37.187692+0800 Demo01_integration SDK[8182:16625633] AVSDK version:1.9.6.47.OpenSDK_1.9.6- 34109
2018-03-27 15:22:37.189444+0800 Demo01_integration SDK[8182:16625633] IMSDK version:v2.5.6.11389.11327
```

You have successfully integrated ILiveSDK.

Email

If you have any questions, send us an email to trtcfb@qq.com.

PC

Last updated：2018-09-28 17:01:18

This document describes how to integrate the TRTC SDK to a PC.

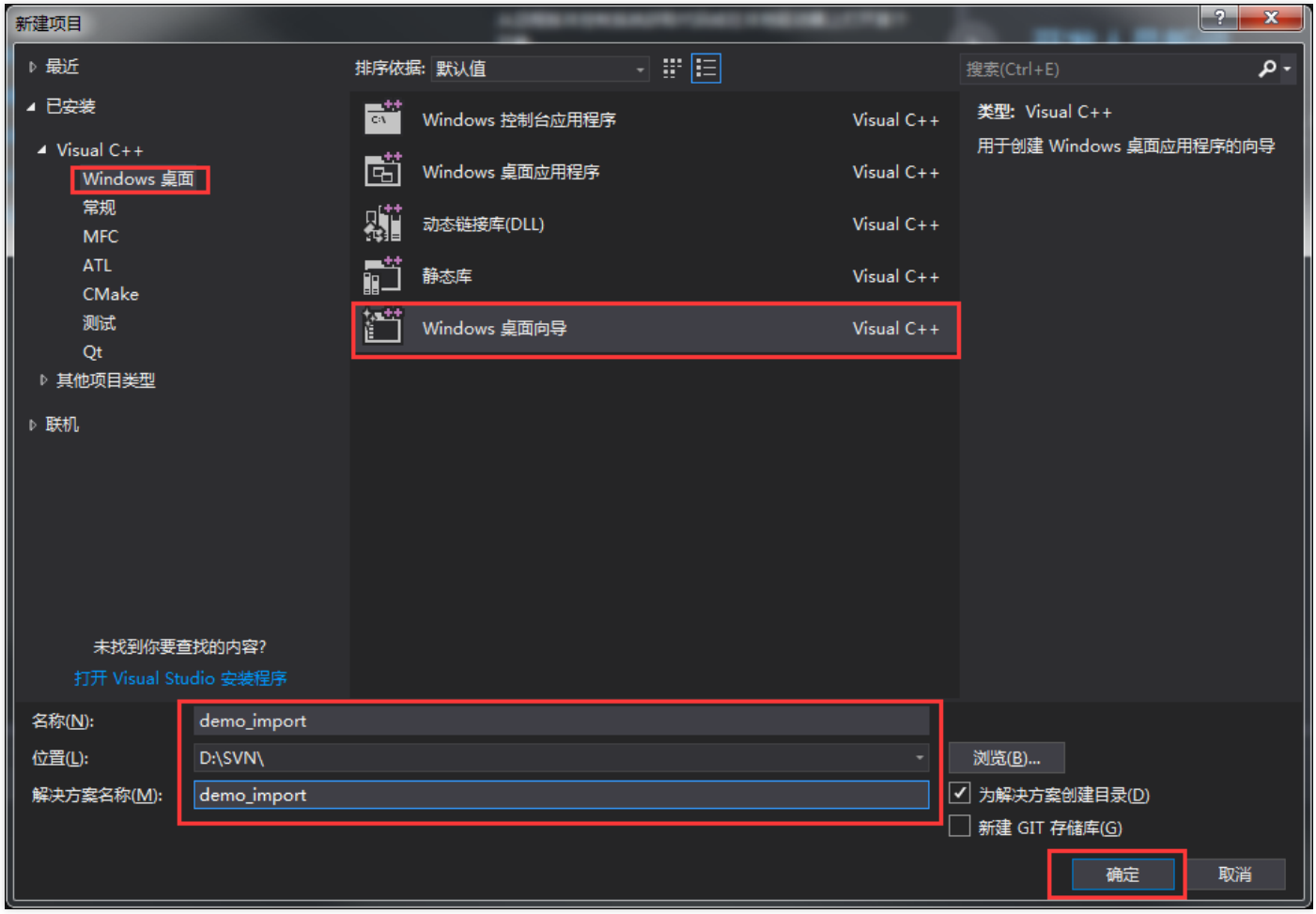
Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

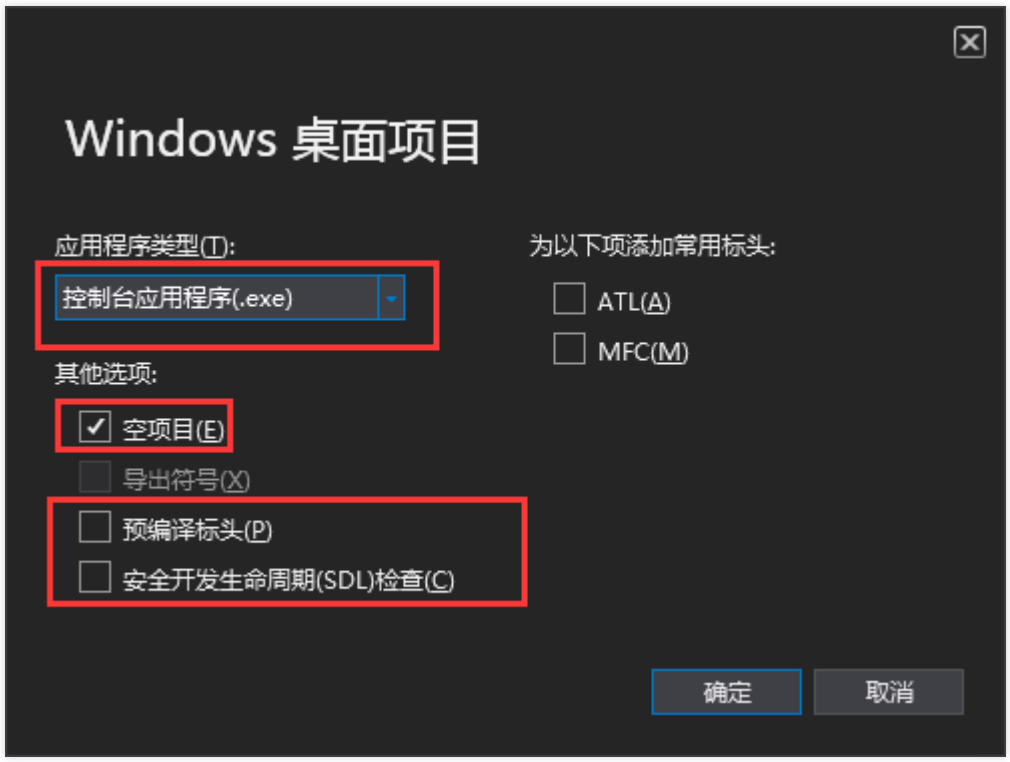
Procedure

Create a Win32 Console project

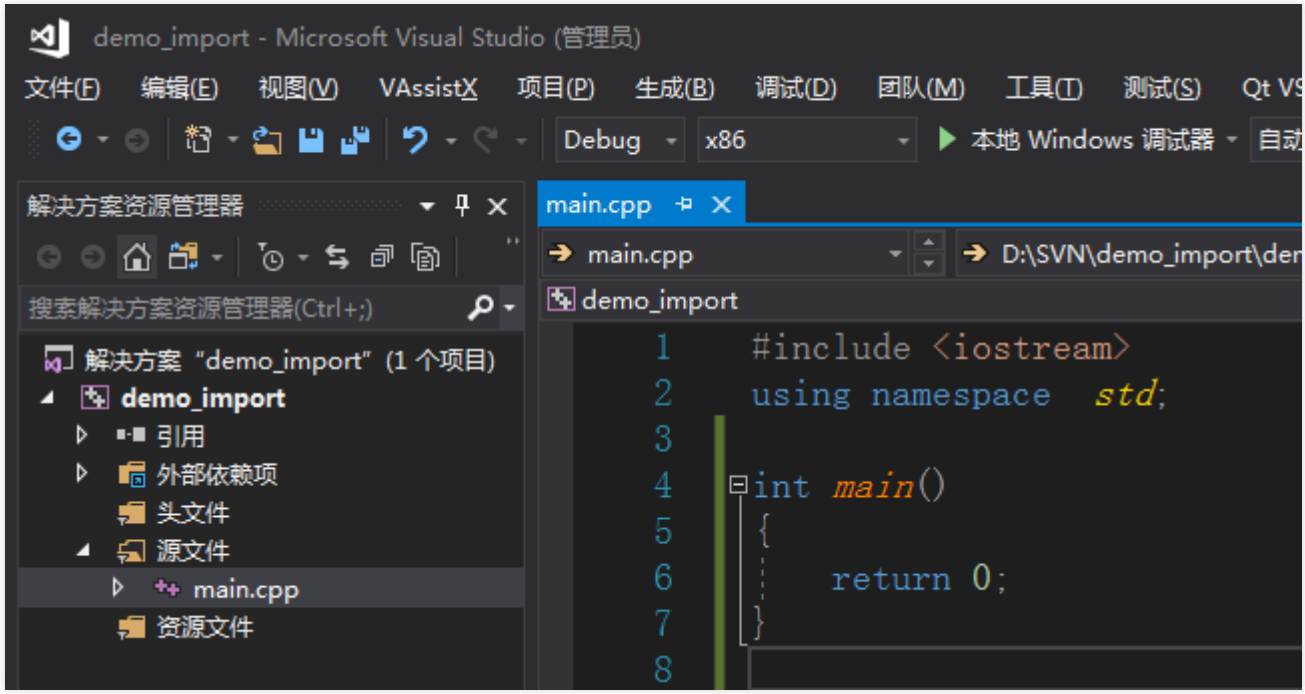
1. Open Visual Studio, click the **File** menu, and then select **New** -> **Project** -> **Create a Project**:



2. Create an empty project:



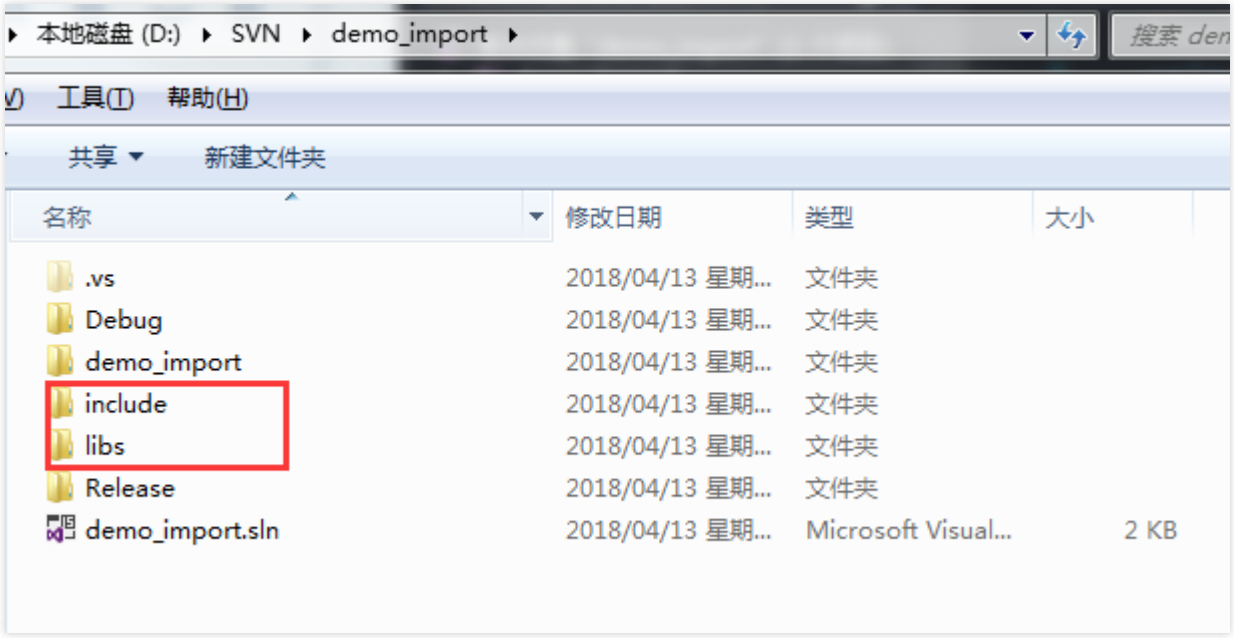
3. Add a cpp file to the project and write an empty main function:



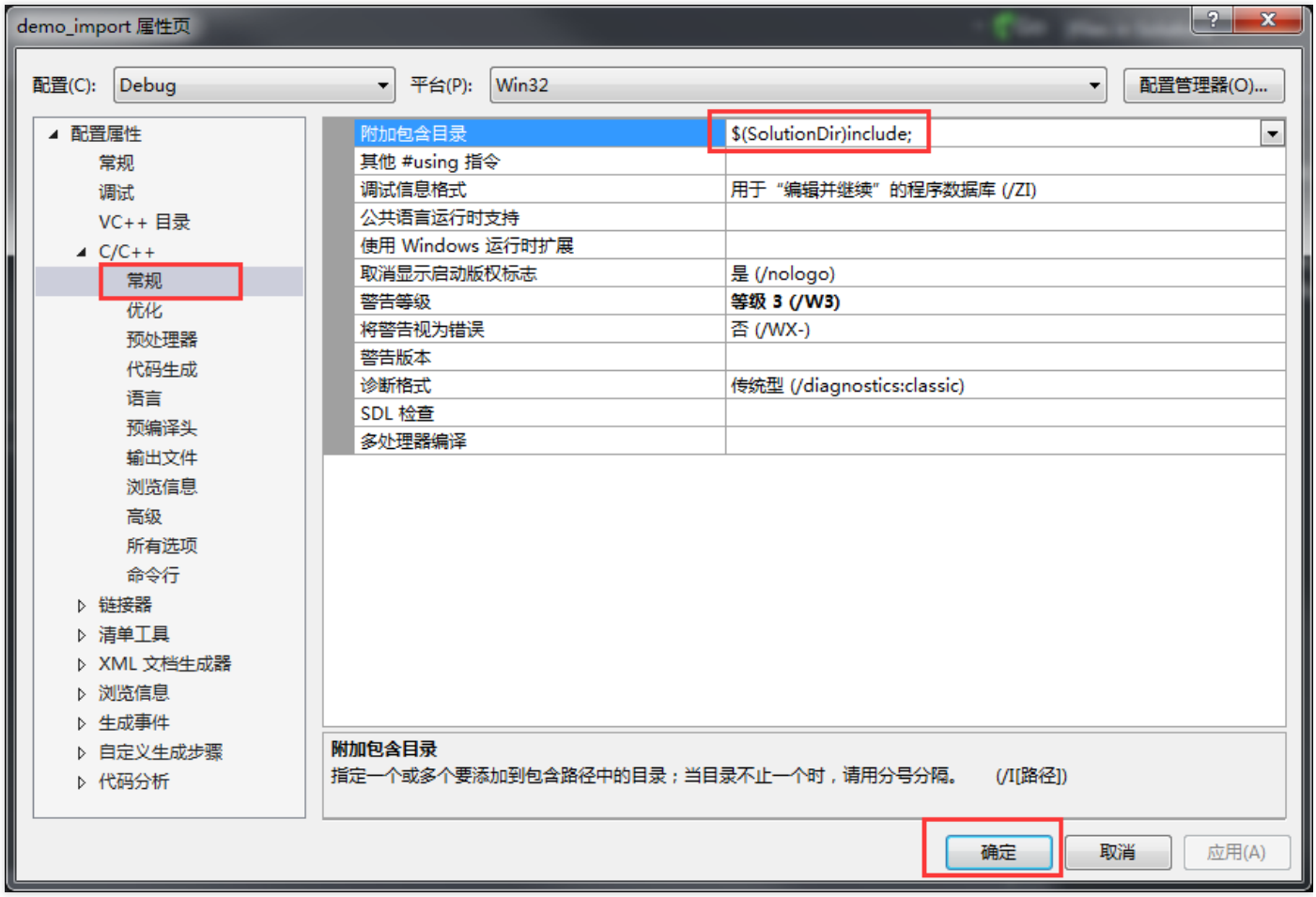
Integrate iLiveSDK

1. Download iLiveSDK
   Download iLiveSDK at Github

2. Copy files
   Copy "include" and "libs" under the iLiveSDK directory to the directory where the solution file ".sln" is located:

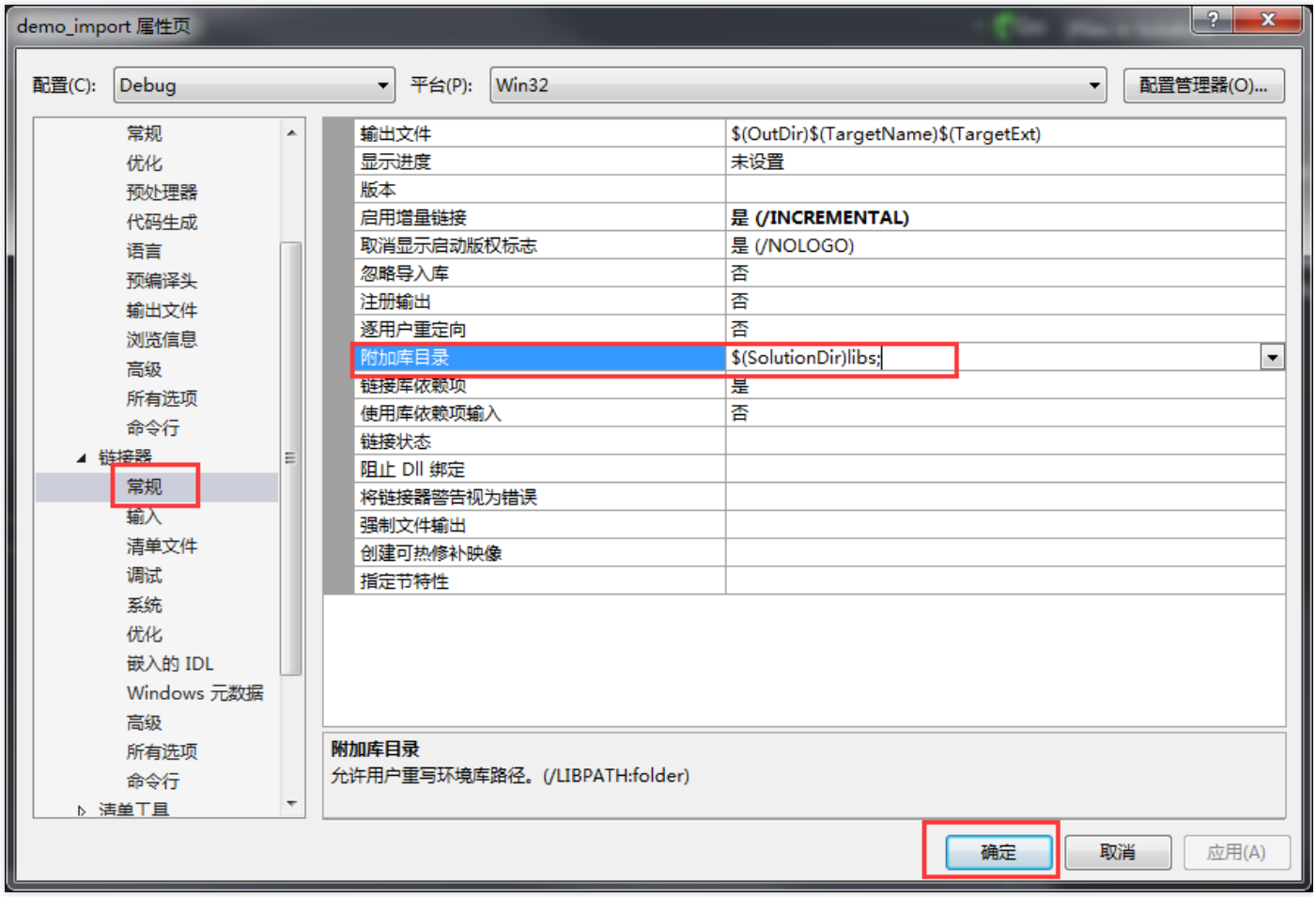


3. Add the "include" directory
   Add `$(SolutionDir)include` to the project's additional include directory:



4. Add the library directory

   Add `$(SolutionDir)libs` to the project's additional library directory:



> **Note:**
> You must configure Debug and Release versions before adding the above include directory and library directory.

5. Include the header file:
   Include the header file in the project and use the ilive namespace to load the lib file of the dynamic library:

```
#include "iLive.h"
using namespace ilive;
#pragma comment(lib, "iLiveSDK.lib")
```

6. Copy the dll file to the directory containing .exe:
   Copy all the dll files in the libs directory to the solution's Debug and Release directories (both

directories will not be generated until Debug and Release are compiled at least once), and then delete the dll files in the libs directory.

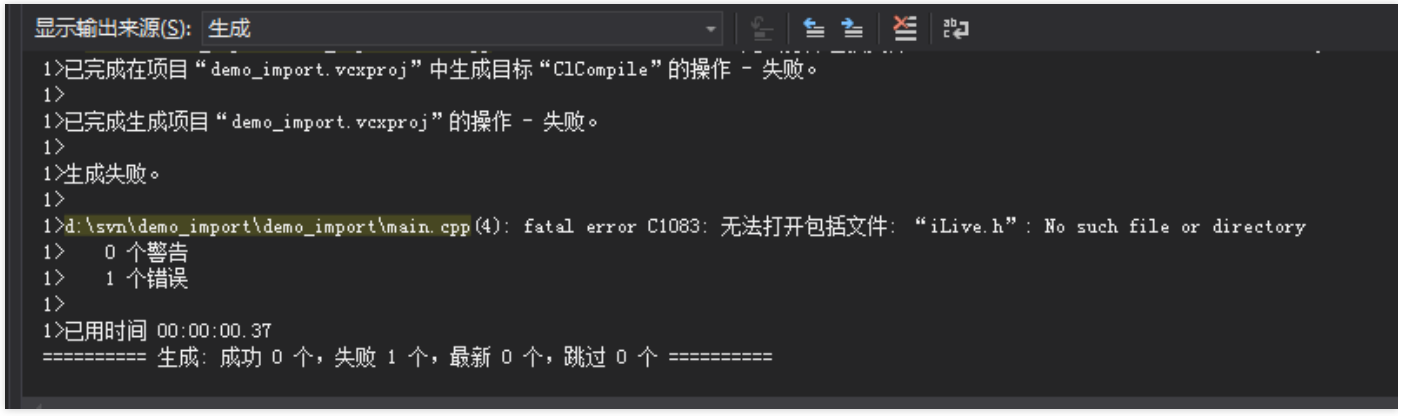
> **Note:**
>] Do not delete iLiveSDK.lib.

7. Verify whether the configuration is successful
   Call GetILive() -> getVersion(). Then the current version number of iLiveSDK is returned.

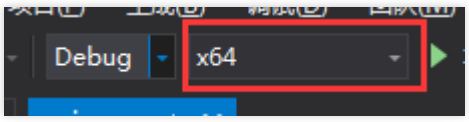
```
cout << GetILive()->getVersion() << endl;
```

Source code description

- Error compiling demo source code:



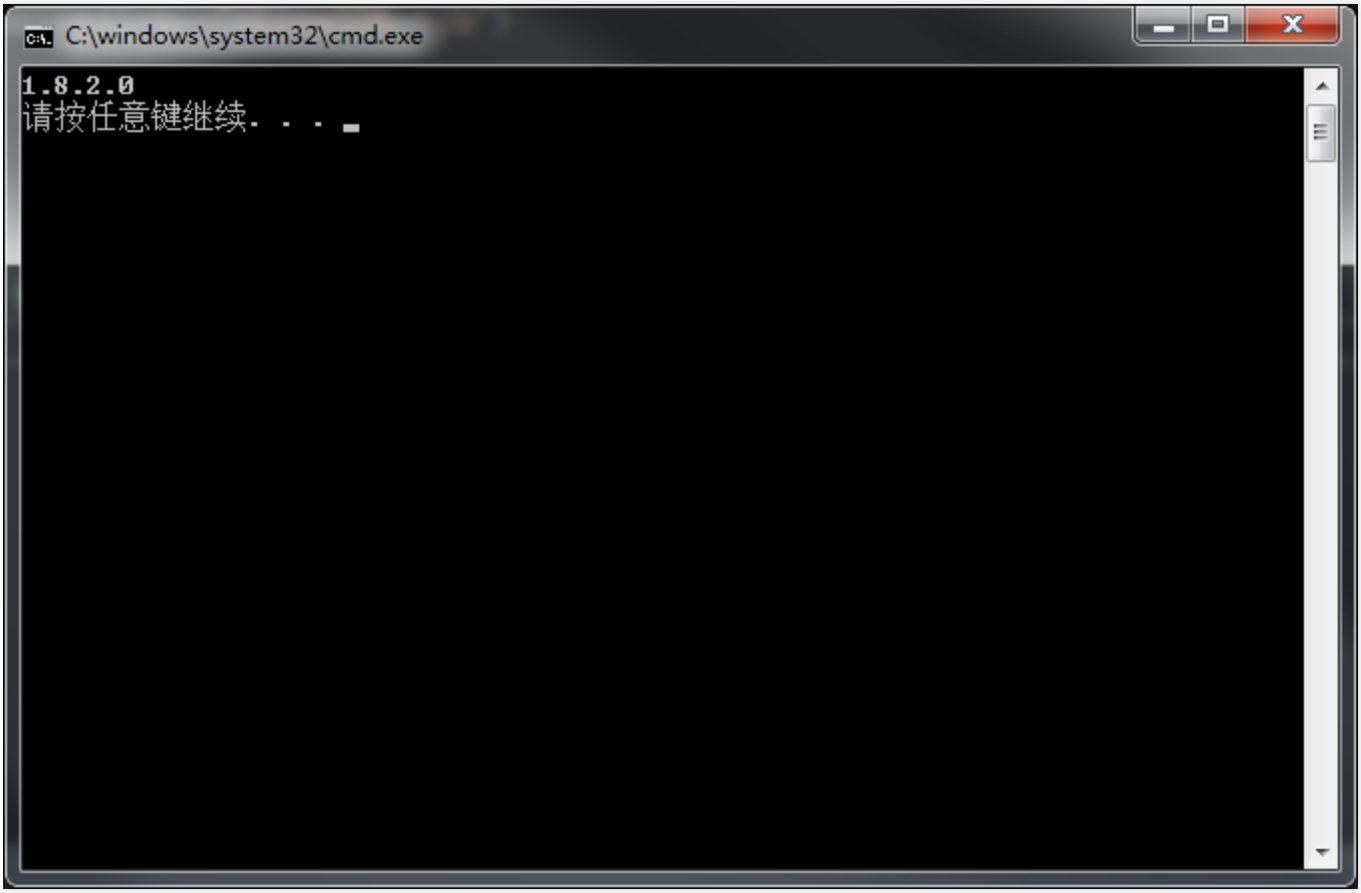
  - Possibly because the 64-bit version is selected in project configuration.



    iLiveSDK is not supported on the 64-bit version. Change it to 32-bit.

Execution results

Press [Ctrl + F5] to run the program, and then the version number displays:



You have successfully integrated iLiveSDK.

Email

If you have any questions, send us an email to trtcfb@qq.com.

Mac

Last updated：2018-09-28 17:00:40

This document describes how to integrate the TRTC SDK to a Mac device.

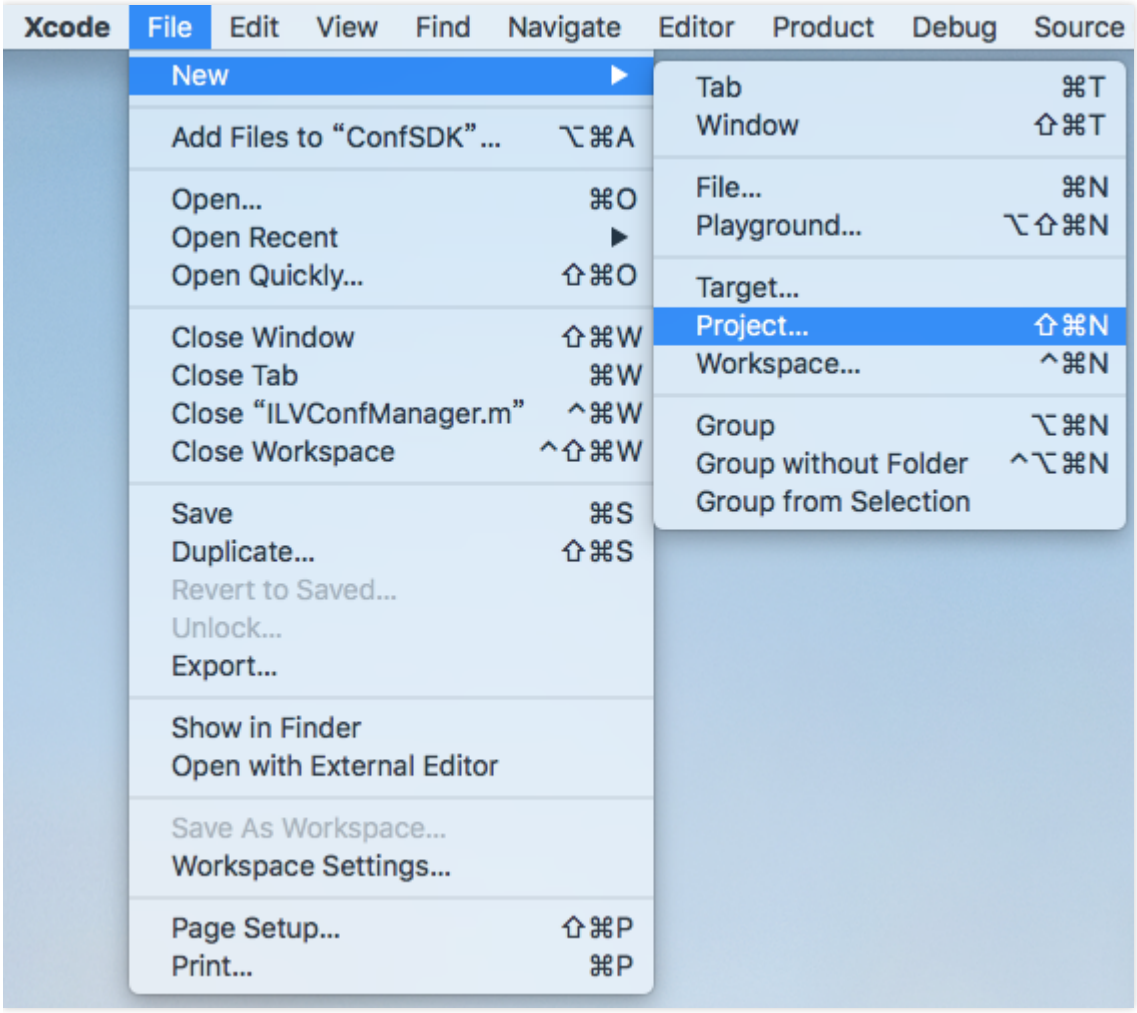
Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

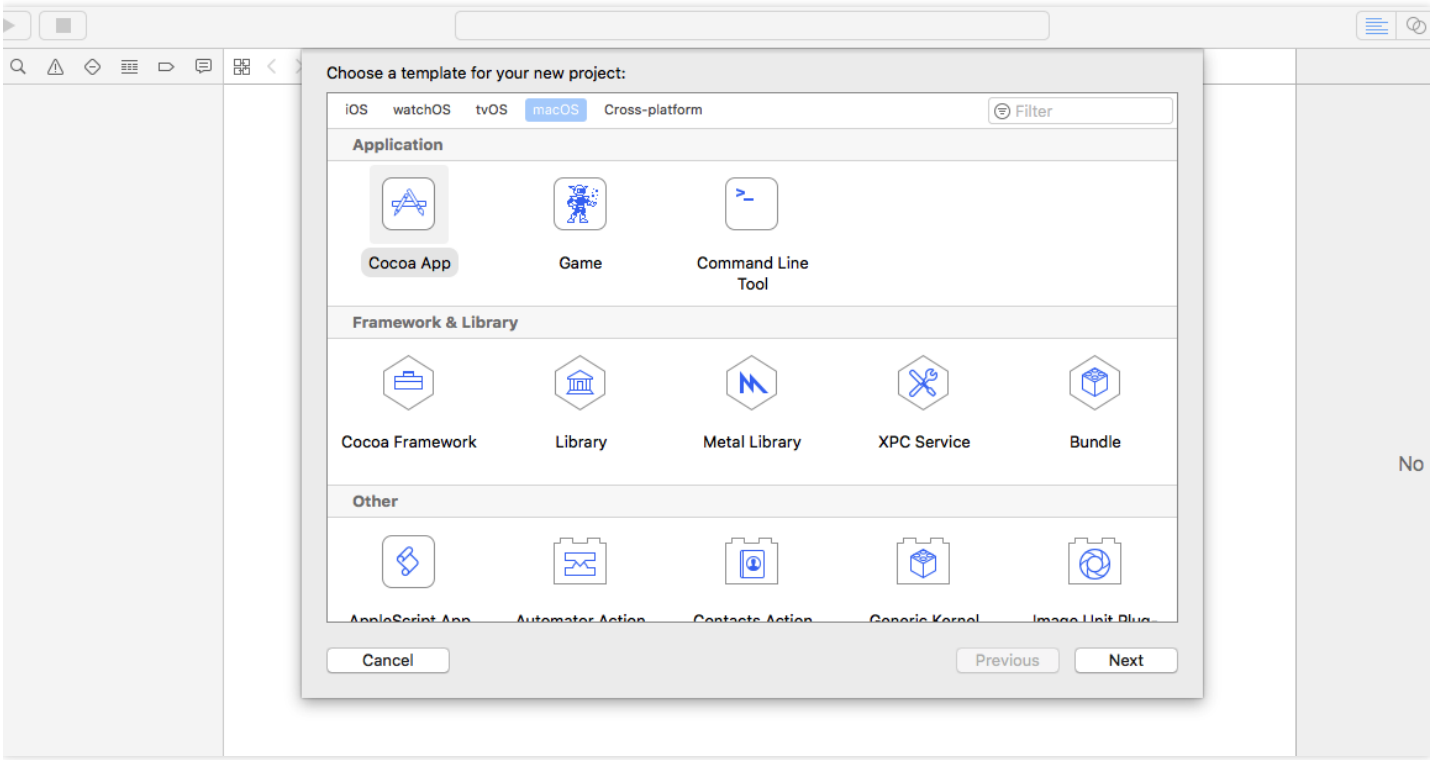
Procedure

Create a Mac project

If you already have a project to integrate, skip to Integrate the SDK.
First, create a project with Xcode to integrate the SDK.
Open Xcode and click **File** -> **New** -> **Project**:



Select **macOS** -> **Cocoa App**



Name the project as TRTCMac Integration SDK and select Objective-C as Language. Enter Team, Organization Name and Organization Identifier as needed. Click **Next**. Select a location to store the project, and then click **Create**.

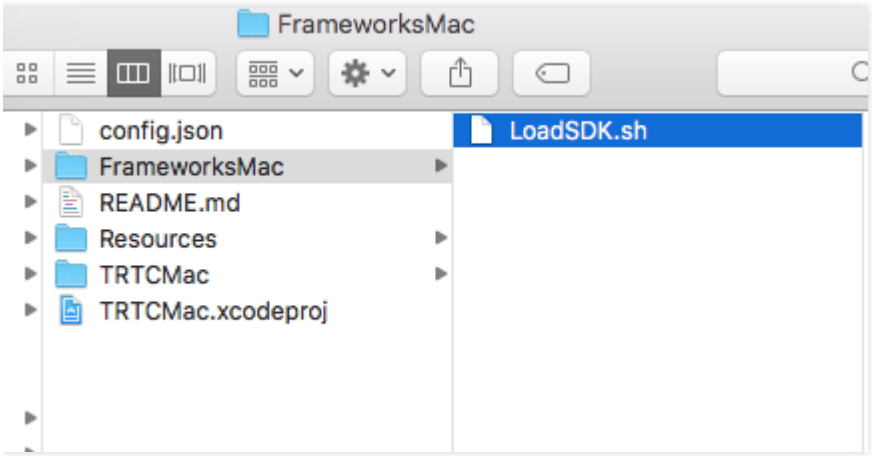
Integrate the SDK

Obtain the SDK

ILiveSDK for Mac is comprised of the following SDKs:

- IMSDK: Provides IM (Instant Messaging)
- AVSDK: Provides the underlying audio/video features
- ILiveSDK: Encapsulates audio/video APIs based on AVSDK to make them easy to use

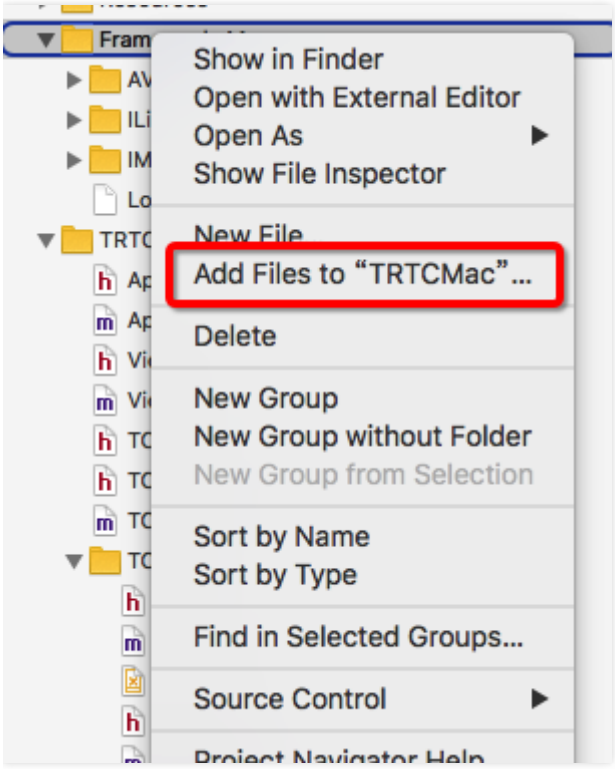
Create a folder named FrameworksMac in the project directory to store the SDK. As ILiveSDK has multiple SDKs, an SDK download script is provided for you to obtain all these SDKs.
Click the download script and place it under the FrameworksMac folder you just created:



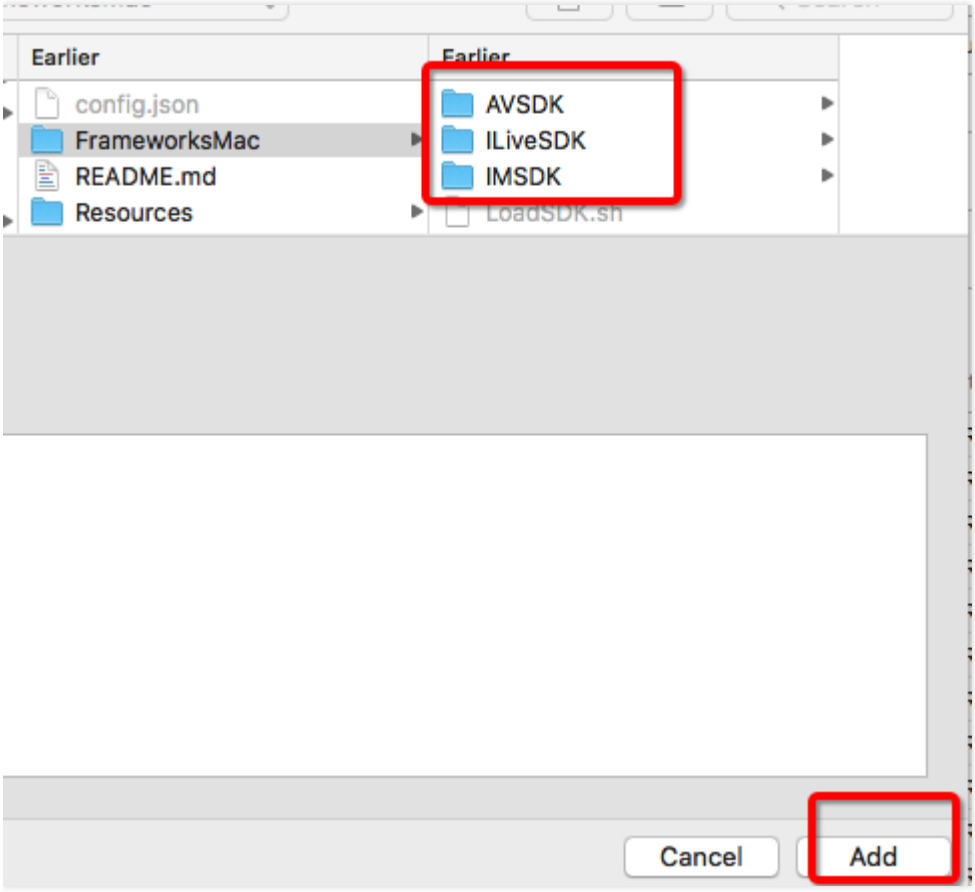
Run the download script (Open the terminal, run the cd command to go to the FrameworksMac directory, and then run sh LoadSDK.sh) to download all the SDKs. After a moment, the downloaded package will be automatically decompressed and deleted.

Import the SDK

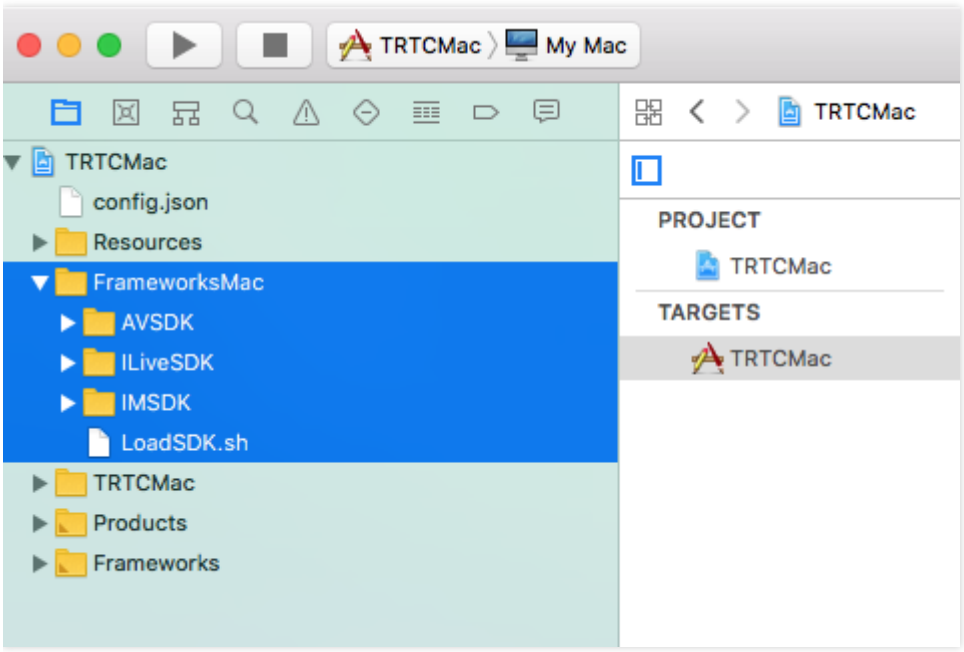
After the download is completed, right-click on **Project** -> **Add Files to "TRTCMac"** to import the SDK to the project:



Select the new **FrameworksMac** folder in the pop-up select box, and then click **Add**:

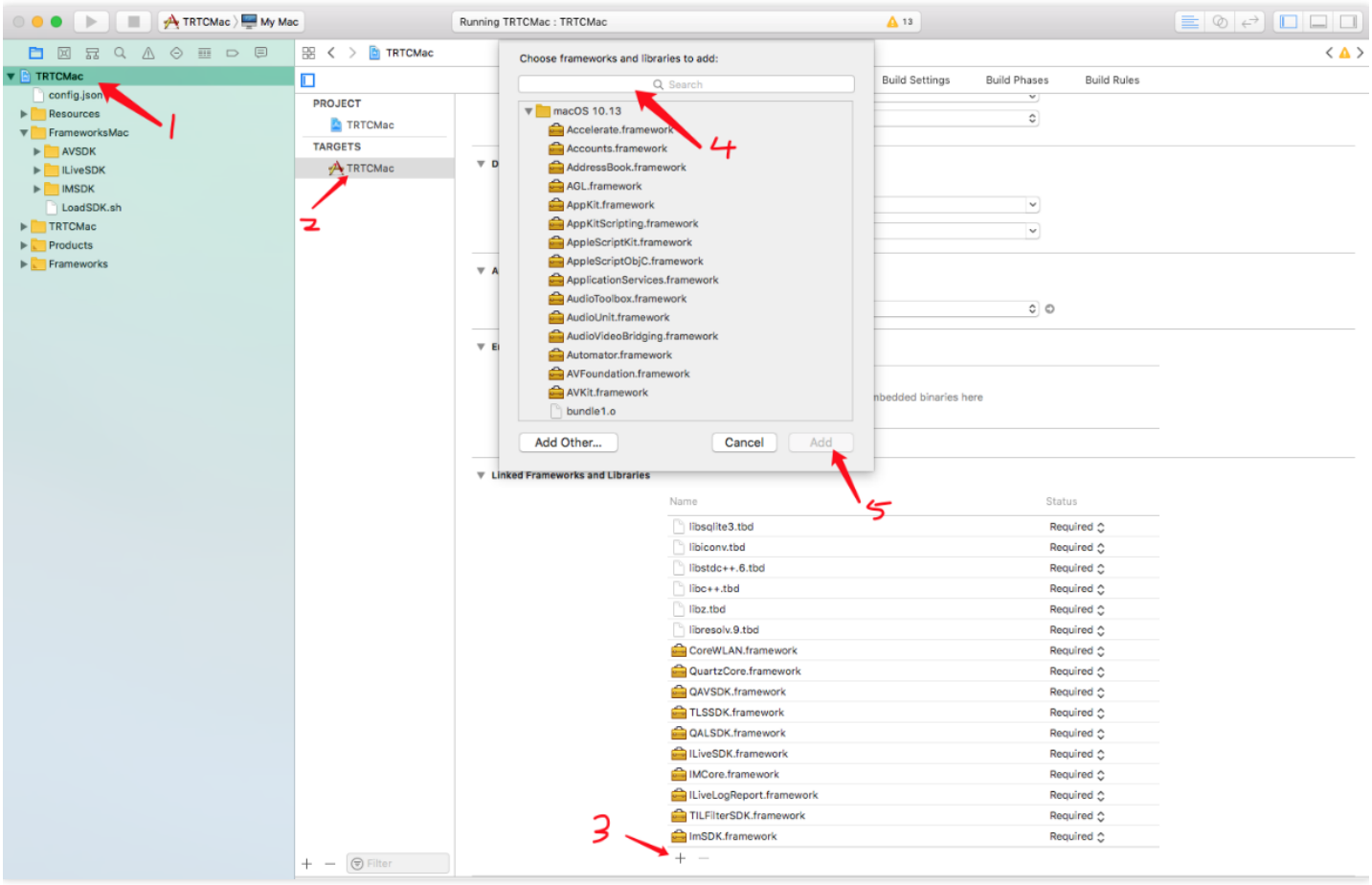


The added project directory is as follows:



Add system dependent libraries

Some system libraries relied on by SDKs in ILiveSDK need to be added to the project.
Click **PROJECT** -> **TARGETS** -> **General**. At the bottom of Linked Frameworks and Libraries section, click **+**, enter the system library name, and click **Add**.



List of system libraries to be added

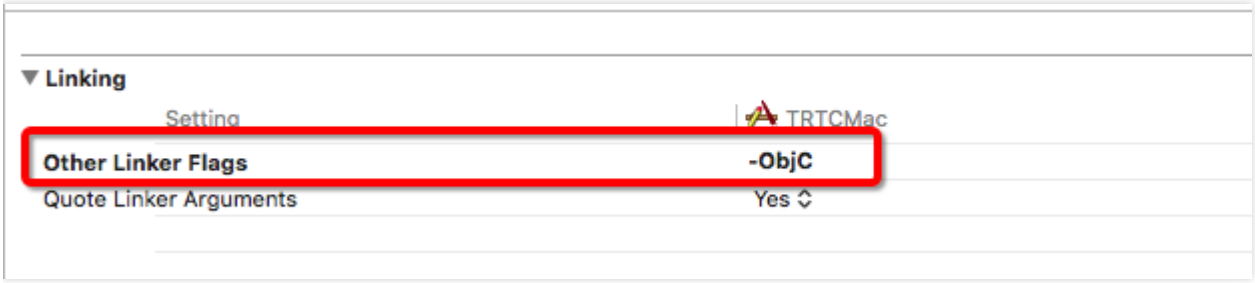
- libsqlite3.tbd
- libiconv.tbd
- libstdc++.6.tbd
- libc++.tbd
- libz.tbd
- libresolve.9.tbd
- CoreWLAN.framework
- QuartzCore.framework
- QAVSDK.framework
- TLSSDK.framework
- QALSDK.framework
- ILiveSDK.framework
- IMCore.framework
- IliveLogReport.framework

- TILFilterSDK.framework
- ImSDK.framework

All the added system libraries are stored in the Frameworks folder. To add these system libraries to your project in an easy manner, you can download our demo code (Click to download) and directly drag the system libraries in the Frameworks folder to the Linked Frameworks and Libraries section under your project.

Configure the project

To use the SDK properly, you also need to make the following configurations:
-Configure ObjC
Build Settings -> **Other Linker Flags** -> **-ObjC**:



Run and check the SDK

After the above steps are completed, you can use ILiveSDK. Add the code to the viewDidLoad function of ViewController.m to obtain the version number:

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

//Obtain the version number
NSLog(@"ILiveSDK version:%@",[[ILiveSDK getInstance] getVersion]);
NSLog(@"AVSDK version:%@",[QAVContext getVersion]);
NSLog(@"IMSDK version:%@",[[TIMManager sharedInstance] GetVersion]);

//Print results
2018-09-03 11:49:28.060945+0800 TRTCMac[73399:23447259] ILiveSDK version:1.9.3.13966
2018-09-03 11:49:28.060956+0800 TRTCMac[73399:23447259] AVSDK version:1.9.9.1012.Local
2018-09-03 11:49:28.060969+0800 TRTCMac[73399:23447259] IMSDK version:v2.5.4.10421.10420
```

You have successfully integrated ILiveSDK.

Email

If you have any questions, send us an email to trtcfb@qq.com.

Log In
Android

Last updated：2018-09-28 17:02:27

This document describes how to integrate iLiveSDK to your client using the repository and how to log in to Tencent Cloud.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Prerequisites

You must have activated the service and created an application at the TRTC official website.

Concepts

- TRTC application
- Application ID(sdkAppId)
- Account type(accountType)
- User ID(userId)
- User signature(userSig)

Obtaining userSig

Each user at Client is provided with a userSig. It is valid for three months upon generation. If the userSig expires, the user cannot log in to TRTC and will receive the error code 8051. At this time, the user should generate a new userSig to log in to TRTC.

```
/** Ticket expired (Need to update ticket userSig) */
public static final int ERR_EXPIRE = 8051;
```

Note: For more information on how to obtain a userSig, see User Authentication.. During debugging, you can directly use the development tools in the console to generate a userSig.

Adding a Dependency (Integrating SDK)

Modify the build.gradle file and add the dependency of iLiveSDK in "dependencies":

```
compile 'com.tencent.ilivesdk:ilivesdk:latest.release' //latest.release refers to the latest iLiveSDK version number
```

Initializing iLiveSDK

Add DemoApp.java and inherit Application:

```
public class DemoApp extends Application {
@Override
public void onCreate() {
super.onCreate();

//Check whether initialization is only performed in the main thread
if (MsfSdkUtils.isMainProcess(this)) {
//Initialize iLiveSDK
ILiveSDK.getInstance().initSdk(this, Constants.SDKAPPID, Constants.ACCOUNTTYPE);
//Initialize iLiveSDK room management module
ILiveRoomManager.getInstance().init(new ILiveRoomConfig());
}
}
}
```

In the application of AndroidManifest.xml, declare:

```
<application
.....
android:name=".DemoApp">
.....
</application>
```

Creating the Login Module

Create a communication API for the login module and Activity:

```
public interface ILoginView {
//Login successful
void onLoginSDKSuccess();
//Login failed
void onLoginSDKFailed(String module, int errCode, String errMsg);
}
```

At the same time, create a LoginHelper.java for login:

```
public class LoginHelper {
private ILoginView loginView;

public LoginHelper(ILoginView view){
loginView = view;
}

public void loginSDK(String userId, String userSig){
ILiveLoginManager.getInstance().iLiveLogin(userId, userSig, new ILiveCallBack() {
@Override
public void onSuccess(Object data) {
loginView.onLoginSuccess();
}

@Override
public void onError(String module, int errCode, String errMsg) {
loginView.onLoginFailed(module, errCode, errMsg);
}
});
}
}
```

Listening for Account Status

The Listening feature of the application is used to listen for forced logout due to repeated login or expired userSig.
You can create an observer to listen globally:

```
/**
* Status observer
*/
public class StatusObservable implements ILiveLoginManager.TILVBStatusListener {

//Message listening linked list
private LinkedList<ILiveLoginManager.TILVBStatusListener> listObservers = new LinkedList<>();
//Handle
private static StatusObservable instance;


public static StatusObservable getInstance(){
if (null == instance){
synchronized (StatusObservable.class){
if (null == instance){
instance = new StatusObservable();
}
}
}
return instance;
}


//Add an observer
public void addObserver(ILiveLoginManager.TILVBStatusListener listener){
if (!listObservers.contains(listener)){
listObservers.add(listener);
}
}

//Remove an observer
public void deleteObserver(ILiveLoginManager.TILVBStatusListener listener){
listObservers.remove(listener);
}

//Obtain the number of observers
public int getObserverCount(){
return listObservers.size();
}

@Override
public void onForceOffline(int error, String message) {
//Copy the linked list
LinkedList<ILiveLoginManager.TILVBStatusListener> tmpList = new LinkedList<>(listObservers);
for (ILiveLoginManager.TILVBStatusListener listener : tmpList){
```

```
listener.onForceOffline(error, message);
}
}
}
```

After successful login, call the API to set listening:

```
ILiveLoginManager.getInstance().setUserStatusListener(StatusObservable.getInstance());
```

In this way, you will know you are forced logout after receiving the OnForceOffline event.

UI Development

A separate login page is required to enter the user name and password at the client. Such basic knowledge of Android development will not be discussed here.

Log in to the onCreated event in the application, and then you can create the module defined above:

```
loginHelper = new LoginHelper(this);
```

Click the login event to obtain the userId and userSig entered by the user. Call the login API of loginHelper to log in:

```
loginHelper.loginSDK(userId, userSig);
```

FAQ

- Failed to download aar

```
Error:Could not resolve all files for configuration ':app:debugCompileClasspath'.
> Could not resolve com.tencent.ilivesdk:ilivesdk:1.8.3.
Required by:
project :app
> Could not resolve com.tencent.ilivesdk:ilivesdk:1.8.3.
> Could not get resource 'https://jcenter.bintray.com/com/tencent/ilivesdk/ilivesdk/1.8.3/ilivesdk-1.8.3.pom'.
> Could not GET 'https://jcenter.bintray.com/com/tencent/ilivesdk/ilivesdk/1.8.3/ilivesdk-1.8.3.pom'.
> Connect to jcenter.bintray.com:443 [jcenter.bintray.com/75.126.118.188] failed: Connection timed out: connect
```

Check the network first. Then, check whether the jcenter website is accessible by clicking on the above link. If a proxy is required, check if it is configured in gradle.properties.

- Log in to IMSDK, a module that does not return any error. Error code: 70009. Error description: tls_check*signature failed decrypt sig failed failed iRet:-2 sdkappid:14000xxxxx,acctype:xxxx,identifier:guest sig:E9vB6Ocs42J8A5lZW6s*

> This may be caused by mismatch between the userSig and ID. Check whether the key for generating the userSig matches the sdkAppId used in initialization.

Email

If you have any questions, send us an email to trtcfb@qq.com.

iOS

Last updated：2018-09-28 17:05:39

This document describes how to initialize the SDK and call the login API.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Prerequisites

You must have activated the service and created an application at the TRTC official website.

Concepts

- TRTC application
- Application ID(sdkAppId)
- Account type(accountType)
- User ID(userId)
- User signature(userSig)

Obtaining userSig

Each user at Client is provided with a userSig. It is valid for three months upon generation. If the userSig expires, the user cannot log in to TRTC and will receive the error code 8051. At this time, the user should generate a new userSig to log in to TRTC.

```
ERR_EXPIRE = 8051, //Ticket expired (Need to update ticket userSig)
```

Note: For more information on how to obtain a userSig, see User Authentication. During debugging, you can directly use the development tools in the console to generate a userSig.

Initializing iLiveSDK

It is recommended to initialize the SDK when you start the application. The SDKAppId and AccountType generated after creation of the application in the Tencent Cloud backend are required for initialization. The sample code is as follows:

```
> AppDelegate.m

//Import header files
#import <ILiveSDK/ILiveSDK.h>

//Define SDKAppId and AccountType
static const int kSDKAppID = SDKAppId generated after creation of the application at backend;
static const int kAccountType = AccountType generated after creation of the application at backen
d;

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)lau
nchOptions {

// Initialize the SDK
[[ILiveSDK getInstance] initSdk:kSDKAppID accountType:kAccountType];

return YES;
}
```

Calling the Login API

The login API is located in the ILiveLoginManager, you need to import it before calling. The calling method is as follows:

```
- (void)iLiveLogin:(NSString *)uid sig:(NSString *)sig succ:(TCIVoidBlock)succ failed:(TCIErrorBlock)fai
led;
```

Enter the userId and userSig generated by the development tools at backend:

```
> ViewController.m

//Import header files
#import <ILiveSDK/ILiveLoginManager.h>
```

```
- (void)viewDidLoad {
[super viewDidLoad];

// Set userId and userSig
self.userIdTF.text = @"use's userId";
self.userSigTF.text = @"usr-generated userSig";
}

//Click the login button
- (IBAction)onLogin:(id)sender {
//Log in to the SDK
[[ILiveLoginManager getInstance] iLiveLogin:self.userIdTF.text sig:self.userSigTF.text succ:^{
NSLog(@"Login successful!") ;
} failed:^(NSString *module, int errId, NSString *errMsg) {
NSLog(@"errId:%d, errMsg:%@",errId, errMsg);
}];
}
```

Listening for User Status

Users may be forcibly logged out due to repeated login or expired userSig. To enable the application to listen for the status, do the following:

Set the calling method for the observer:

```
[[ILiveSDK getInstance] setUserStatusListener:self];
```

The observer complies with the TIMUserStatusListener protocol and implements the following methods:

```
/**
* Notification of forced logout
*/
- (void)onForceOffline;

/**
* The userSig expired (Obtain a new userSig to log in)
*/
- (void)onUserSigExpired;
```

Note:
In the sample application, the storyboard is used to build a simple interface. In practical

applications, customers need to place the login code in the right place according to their own business logic.
The above code is only for demonstration purposes. In practice, a customer has an independent account. So the login process is as follows:

Enter your user name and password on the login page displayed on the client. After your account is verified, a userSig is generated and returned to the client via the TRTC SDK. The client uses the user name and userSig to call the login API of ILiveSDK.

Run the application. When a prompt saying "Login successful" appears, you have successfully logged in to ILiveSDK.

Email

If you have any questions, send us an email to trtcfb@qq.com.

PC

Last updated：2018-09-28 17:03:50

This document describes how to log in to Tencent Cloud from your client.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Prerequisites

You must have activated the service and created an application at the TRTC official website.

Concepts

- TRTC application
- Application ID(sdkAppId)
- Account type(accountType)
- User ID(userId)
- User signature(userSig)

Obtaining a userSig

For more information on how to obtain a userSig, see User Authentication.
You can also use the development tools to generate a userSig.

Initializing iLiveSDK

Before login, it is recommended to initialize the SDK when you run the application. The sample code is as follows:

```
#define SDKAppId SDKAppId generated after creation of the application at backend
#define AccountType AccountType generated after creation of the application at backend
```

```
int nRet = GetILive()->init(SDKAppId, AccountType, false);
if (nRet != NO_ERR)
{
//Initialization failed. nRet is the error code
}
else
{
//Initialization successful
}
```

Calling the Login API

You can log in after initialization. The sample code is as follows:

```
GetILive()->login(userId, UserSig, [](void* data) {
//Login successful
}, [](const int code, const char *desc, void* data) {
//Login failed. "code" is the error code and "desc" is the error description
}, NULL);
```

Source Code Description

- C++11 is used in the demonstration code
  The lambda expression in C++11 is used by the above login API to pass in the successful/failed SDK callback. If you are using vs2010 or an earlier version that is not fully compatible with C++11, the function pointer in C programming language should be passed in for the successful/failed callback, as shown below:

```
void OnLoginSuc(void* data)
{
//Login successful
}

void OnLoginErr(const int code, const char *desc, void* data)
{
//Login failed. "code" is the error code and "desc" is the error description
}
```

```
GetILive()->login(userId, UserSig, OnLoginSuc, OnLoginErr, NULL);
```

The lambda expression is always used.

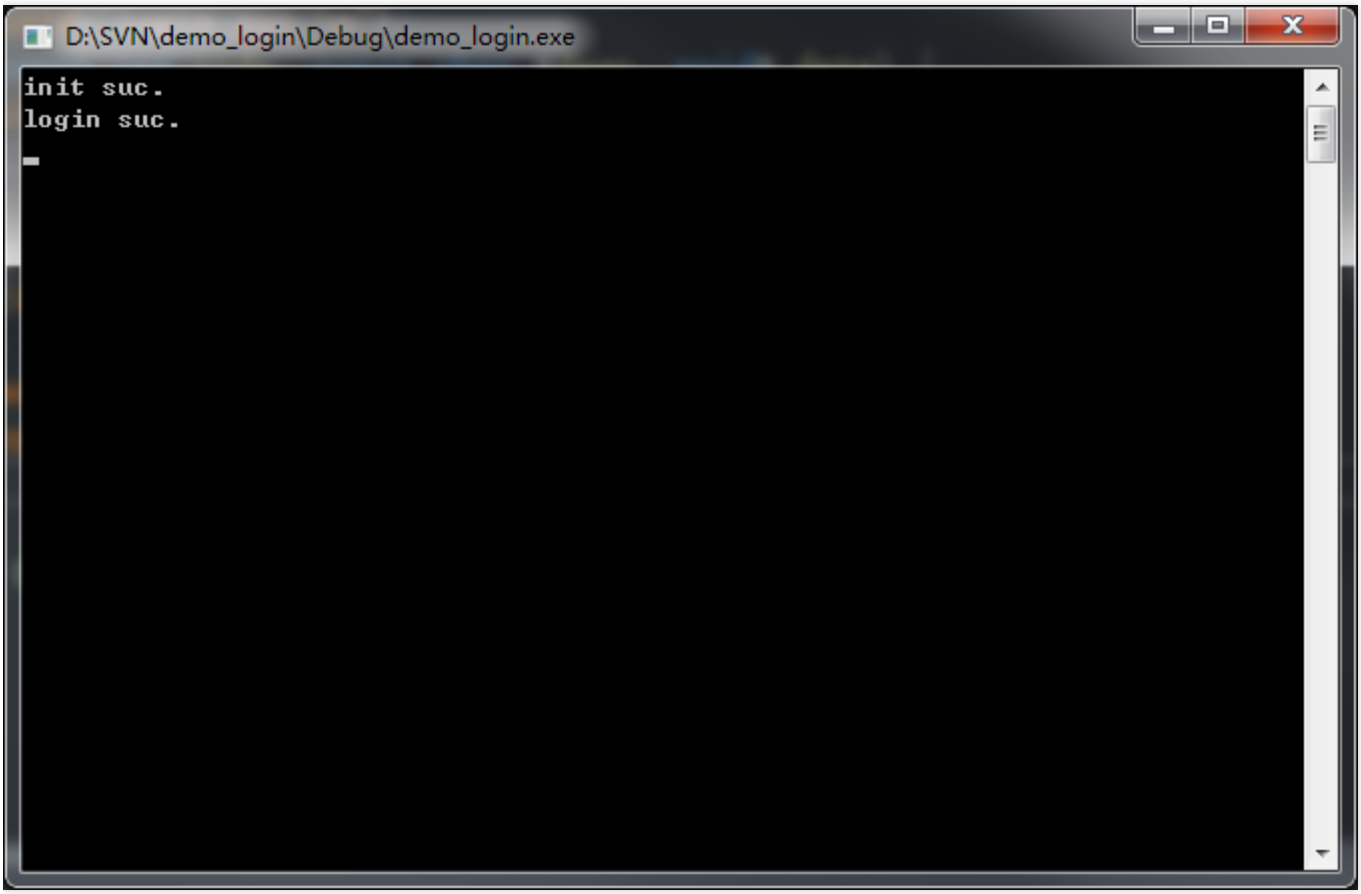
- Each asynchronous callback has a parameter void* data
  As the function pointer in C is used for callback, variables outside of the callback function cannot be accessed during callback. During asynchronous callback, pass in the parameter void*. The parameter will not change when passed out. Pass in the pointer "this" of the class object, forcibly convert it to the pointer of the class object during callback, and then access the class members via this pointer.

- iLiveSDK relies on Windows message loop
  The iLiveSDK SDK uses Windows message loop to throw callbacks back to the main thread (GUI thread) so that users can perform UI operations in callbacks. Therefore, if it is a Win32 console program, there must be a message loop in the program. The code is as follows:

```
#include <Windows.h>
MSG msg;
while (GetMessage(&msg, NULL, 0, 0))
{
TranslateMessage(&msg);
DispatchMessage(&msg);
}
```

- In this demo, userId and userSig are hard-coded. userSig is valid for 3 months. When it expires, the login will fail. When you run the demo, generate your userSig by following the steps described in this document, and then use your SDKAppId, AccountType, userId and userSig for testing.

Execution Results



Email

If you have any questions, send us an email to trtcfb@qq.com.

Mac

Last updated：2018-09-28 17:03:16

This document describes how to initialize the SDK and call the login API.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Prerequisites

You must have activated the service and created an application at the TRTC official website.

Concepts

- TRTC application
- Application ID(sdkAppId)
- Account type(accountType)
- User ID(userId)
- User signature(userSig)

Obtaining userSig

Each user at Client is provided with a userSig. It is valid for three months upon generation. If the userSig expires, the user cannot log in to TRTC and will receive the error code 8051. At this time, the user should generate a new userSig to log in to TRTC.

```
ERR_EXPIRE = 8051, //Ticket expired (Need to update ticket userSig)
```

Note: For more information on how to obtain a userSig, see User Authentication. During debugging, you can directly use the development tools in the console to generate a userSig.

Initializing iLiveSDK

It is recommended to initialize the SDK when you start the application. The SDKAppId generated after creation of the application in the Tencent Cloud backend is required for initialization. SDKAppId is configured locally in this example, so you can obtain it by reading the configuration. The sample code is as follows:

```
> AppDelegate.m

//Import header files
#import <ILiveSDK/ILiveSDK.h>


- (void)applicationDidFinishLaunching:(NSNotification *)aNotification {

[[ILiveSDK getInstance] initSdk:[TCLiveConfigInfo getInstance].sdkAppID];

}
```

Calling the Login API

The login API is located in the ILiveLoginManager, you need to import it before calling. The calling method is as follows:

```
- (void)iLiveLogin:(NSString *)uid sig:(NSString *)sig succ:(TCIVoidBlock)succ failed:(TCIErrorBlock)fai
led;
```

Enter the userId and userSig generated by the development tools at backend:

```
> ViewController.m

//Import header files
#import <ILiveSDK/ILiveLoginManager.h>

- (IBAction)onLogin:(id)sender {
if ([self.inputTextField stringValue].length >0) {
NSDictionary *dic = [[TCLiveConfigInfo getInstance] parseLoginInfoConfig];
for (NSDictionary *user in dic[@"users"]) {
NSString *userid = user[@"userId"];
NSInteger selectIndex = [self.userName indexOfSelectedItem];
```

```
NSString *selectName = [self.userName itemTitleAtIndex:selectIndex];
if ([userid isEqualToString:selectName]) {
[TCLiveConfigInfo getInstance].userToken = user[@"userToken"];
[TCLiveConfigInfo getInstance].userID = selectName;
break;
}
}
//Step 2: Log in
[[ILiveLoginManager getInstance] iLiveLogin:[TCLiveConfigInfo getInstance].userID sig:[TCLiveConfigI
nfo getInstance].userToken succ:^{
NSLog(@"-----> login succ");
} failed:^(NSString *module, int errId, NSString *errMsg) {

NSLog(@"-----> login fail,%@ %d %@",module, errId, errMsg);
}];

NSString *defaultRole = [[TCLiveConfigInfo getInstance].roles firstObject][@"name"];
_vc = [[TCLiveRoomWC alloc] initWithRoomID:[self.inputTextField stringValue] role:defaultRole];
[_vc.window orderFront:nil];
}
else{

}

}
```

Listening for User Status

Users may be forcibly logged out due to repeated login or expired userSig. To enable the application to listen for the status, do the following:

Set the calling method for the observer:

```
[[ILiveSDK getInstance] setUserStatusListener:self];
```

The observer complies with the TIMUserStatusListener protocol and implements the following methods:

```
/**
* Notification of forced logout
*/
- (void)onForceOffline;

/**
```

```
* The userSig expired (Obtain a new userSig to log in)
*/
- (void)onUserSigExpired;
```

Note:
In the sample application, the storyboard is used to build a simple interface. In practical applications, customers need to place the login code in the right place according to their own business logic.
The above code is only for demonstration purposes. In practice, a customer has an independent account. So the login process is as follows:

Enter your user name and password on the login page displayed on the client. After your account is verified, a userSig is generated and returned to the client via the TRTC SDK. The client uses the user name and userSig to call the login API of ILiveSDK.

Run the application. When a prompt saying "Login successful" appears, you have successfully logged in to ILiveSDK.

Email

If you have any questions, send us an email to trtcfb@qq.com.

Create and Enter a Room Android

Last updated：2018-09-28 17:08:23

This document describes how to create a room in the client and publish a live video.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Concepts

- Room
- privateMapKey
- Role configuration
- Rendering control
  You need a platform to display the obtained video data. That is the rendering control, which can correspond to an Android control.

Adding Rendering Control

You must first add a control to the previous demo layout to render the video:

```
<com.tencent.ilivesdk.view.AVRootView
android:id="@+id/av_root_view"
android:layout_width="match_parent"
android:layout_height="match_parent" />
```

Creating a Room Module

Define an API for the communication between the room module and Activity:

```
public interface IRoomView {
// Entered the room successfully
void onEnterRoom();
// Failed to enter the room
void onEnterRoomFailed(String module, int errCode, String errMsg);

// Exited room successfully
void onQuitRoomSuccess();
// Failed to exit the room
void onQuitRoomFailed(String module, int errCode, String errMsg);

// Room disconnected
void onRoomDisconnect(String module, int errCode, String errMsg);
}
```

Create a room module:

```
public class RoomHelper implements ILiveRoomOption.onExceptionListener, ILiveRoomOption.o
nRoomDisconnectListener {
private IRoomView roomView;

public RoomHelper(IRoomView view){
roomView = view;
}
// Set the rendering control
public void setRootView(AVRootView avRootView){
ILiveRoomManager.getInstance().initAvRootView(avRootView);
}
// Create a room
public int createRoom(int roomId){
ILiveRoomOption option = new ILiveRoomOption()
.privateMapKey(privateMapKey) //Room ticket
.imsupport(false) // Do not need the IM feature
.exceptionListener(this) //Listen on exceptional events
.roomDisconnectListener(this) //Listen on room disconnection events
.controlRole("user") // Use the user role
.autoCamera(true) // Enable the camera automatically and perform the upstream
.autoMic(true); // Enable the microphone automatically and perform the upstream

return ILiveRoomManager.getInstance().createRoom(roomId, option, new ILiveCallBack() {
@Override
public void onSuccess(Object data) {
roomView.onEnterRoom();
}
```

```
@Override
public void onError(String module, int errCode, String errMsg) {
roomView.onEnterRoomFailed(module, errCode, errMsg);
}
});
}
// Exit the room
public int quitRoom(){
return ILiveRoomManager.getInstance().quitRoom(new ILiveCallBack() {
@Override
public void onSuccess(Object data) {
roomView.onQuitRoomSuccess();
}

@Override
public void onError(String module, int errCode, String errMsg) {
roomView.onQuitRoomFailed(module, errCode, errMsg);
}
});
}

// Handle the Activity event
public void onPause(){
ILiveRoomManager.getInstance().onPause();
}
public void onResume(){
ILiveRoomManager.getInstance().onResume();
}

@Override
public void onException(int exceptionId, int errCode, String errMsg) {
//Handle the exceptional event
}

@Override
public void onRoomDisconnect(int errCode, String errMsg) {
//Handle room disconnection (generally, the room is reclaimed in Tencent QCloud server due to net
work outage or no upstream for a long time)
}
}
```

UI Development

In the onCreate event of the Activity of a room, you can create a room module and configure a rendering control.

```
roomHelper = new RoomHelper(this);
// Obtain the rendering control
AVRootView avRootView = findViewById(R.id.av_root_view);
// Set the background color to blue when there is no rendering (Note: you cannot set it directly in th
e layout)
avRootView.getVideoGroup().setBackgroundColor(Color.BLUE);
// Set the rendering control
roomHelper.setRootView(avRootView);
```

You can enter the room number and create a room according to the entered room number in the interface, or the room number can be hard-coded (for testing).

```
roomHelper.createRoom(1234);
```

If the onEnterRoom event is thrown and you can see your video images, the room is created successfully.

FAQ

Error code 10004 Failed to enter a room and prompts "request room server address failed"

Make sure that the room ticket field (privateMapKey) is configured correctly.

> The privateMapKey field is required for new users, and existing users (do not need room tickets) need to configure the field at initialization
>
> ```
> ILiveSDK.getInstance().setChannelMode(CommonConstants.E_ChannelMode.E_ChannelIMSD
> K);
> ```

Error code 71 Failed to enter a room and prompts "decodeSsoCmd_pbvideoapp_pbvideoinfoErr:user id error longConnHead.account=0"

This is caused by login of multiple accounts. Confirm whether the previous account is logged out before you log in to the new account.

Receive EXCEPTION_ENABLE_CAMERA_FAILED in onException and errCode is 1. Failed to enable the camera.

1. Confirm whether Android devices have a camera and the camera is working properly.

2. Confirm whether no other app occupies the camera in the background.
3. For devices with Android 6.0 or above, confirm whether the dynamic permission `Manifest.permission.CAMERA` of the camera is enabled.

Callback for failed operations. Error code 1003 or 8011

1. Operations of entering/exiting rooms are linearly exclusive. If your request is too frequent, SDK may throw 8011, which means the last operation must be completed (callback and return) before continuing (to enter/exit the room);
2. You can only enter one room at a time. If you do not exit the last room before creating (or entering) a new room, 1003 is thrown. In this case, you must exit the last room first.

Email

If you have any questions, send us an email to trtcfb@qq.com.

iOS

Last updated：2018-09-28 17:10:43

This document describes how to create a room, obtain LVB video images and exit the room when appropriate.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Concepts

- Room
- privateMapKey
- Role configuration
- Rendering control
  You need a platform to display the obtained video data. That is the rendering control.
- Video data types
  Tencent Cloud supports performing the upstream of one mainstream and one substream simultaneously in one account. Videos from different sources of video streams are classified into the following types: camera, screen sharing, and video playing (generated from PC).

Video Type	Stream Type	Description
Camera	Mainstream	Data collected from the camera
Screen sharing	Substream	Generated from screen sharing
Video playing	Substream	Generated from video file playing

Procedure

Creating a room

A room can be created by `ILiveRoomManager.h`, which requires two key parameters, room ID (roomId) and room configuration object (option):

- The roomId can be entered by users in an interface (see the demo), or hard-coded in codes for testing. The roomId must conform to the rules described in Preliminary Information.

- The room configuration object must be created by users. The ILiveRoomOption is a class used to configure the instant messaging and other features of the audios and videos in the room to be created. You can use defaultHostLiveOption by default.

The result of creating a room is returned by calling back Block. You can process it in successful/failed callback based on your business logic.

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

//Create a room
- (IBAction)onCreateRoom:(id)sender {
// 1. Create a live room page
LiveRoomViewController *liveRoomVC = [[LiveRoomViewController alloc] init];

// 2. Create a room configuration object
ILiveRoomOption *option = [ILiveRoomOption defaultHostLiveOption];
// Configure a room ticket
option.privateMapKey = privateMapKey;
option.imOption.imSupport = NO;
// Set audio/video listening in the room
option.memberStatusListener = liveRoomVC;
// Set room disconnection event listening
option.roomDisconnectListener = liveRoomVC;

// This parameter indicates the audio/video specification used after a user enters a room. The value o
f the parameter is the role name configured by the customer in **Screen Setting** on the Tencent Clo
ud TRTC console (for example, the default role name is User, and you can set controlRole = @"user")
option.controlRole = #The role name configured on the Tencent Cloud console#;

// 3. Call the API to create a room and pass the roomId and option
[[ILiveRoomManager getInstance] createRoom:[self.roomIDTF.text intValue] option:option succ:^{
// Room created successfully. The user will be redirected to the room page
[self.navigationController pushViewController:liveRoomVC animated:YES];
} failed:^(NSString *module, int errId, NSString *errMsg) {
// Failed to create a room
```

```
NSLog(@"Failed to create room errId:%d errMsg:%@",errId, errMsg);
}];
```

After you enter the room, the camera and microphone are automatically enabled and the stream is automatically pushed, which can be set in the `ILiveRoomOption` configuration object passed when you create the room). In the live room, all audio/video events are notified to the listener through audio/video event callback. First, you must set a listener.

The listener refers to the attribute `memberStatusListener` of the `ILiveRoomOption` object passed when the class is created. Because the room page object `LiveRoomViewController` is set as the listener, the `option` is passed to the `LiveRoomViewController` automatically through the `initWithOption` initialization when the room object is created.

```
> LiveRoomViewController.m

- (instancetype)initWithOption:(ILiveRoomOption *)option {
self = [super init];
if (self) {
option.memberStatusListener = self;
}
return self;
}
```

Listening events in the room

When creating a room, you can set room audio/video event listening and room disconnection event listening in the configuration object to listen for events in the room.
The listener of audio/video events complies with the `ILiveMemStatusListener` protocol and implements the following methods:

```
@protocol ILiveMemStatusListener <NSObject>
/**
The function of the notification of status changes of members in the room, through which the busine
ss side will be notified when the status members in the room change (such as publishing audios or vi
deos).

@param event Status change ID. For more information, see the definition of QAVUpdateEvent
@param endpoints List of member IDs whose statuses change .

@return YES Operation succeeded
*/
- (BOOL)onEndpointsUpdateInfo:(QAVUpdateEvent)event updateList:(NSArray *)endpoints;
@end
```

Notes:
This method is a callback method for audio/video events in the room. If someone in the room enables the camera and microphone, the underlying SDK will call back this method to notify the listener. This method has two parameters, event and endpoints.

- The event is an enumerated value for events, which defines the types of events that could occur to members in the room, including entering and exiting the room, and enabling/disabling cameras and microphones.
- The endpoints is an array of QAVEndpoint objects, which specifies each user sending an event. This method is used to send notifications of specific types of changes in audios and videos and the users who make those changes. When an event change is listened for, some adjustments must be made on the interface. For example, when it finds that a user camera is enabled, a rendering image should be added to the interface to render the user's image.

The event has the following values:

```
typedef NS_ENUM(NSInteger, QAVUpdateEvent) {
QAV_EVENT_ID_NONE = 0, ///< Default value. Ignored..
QAV_EVENT_ID_ENDPOINT_ENTER = 1, ///< Event of entering a room..
QAV_EVENT_ID_ENDPOINT_EXIT = 2, ///< Event of exiting a room..
QAV_EVENT_ID_ENDPOINT_HAS_CAMERA_VIDEO = 3, ///< Has a camera video event.
QAV_EVENT_ID_ENDPOINT_NO_CAMERA_VIDEO = 4, ///< No camera video event.
QAV_EVENT_ID_ENDPOINT_HAS_AUDIO = 5, ///< Has an audio event.
QAV_EVENT_ID_ENDPOINT_NO_AUDIO = 6, ///< No audio event.
QAV_EVENT_ID_ENDPOINT_HAS_SCREEN_VIDEO = 7, ///< Has a screen video event.
QAV_EVENT_ID_ENDPOINT_NO_SCREEN_VIDEO = 8, ///< No screen video event.
QAV_EVENT_ID_ENDPOINT_HAS_MEDIA_FILE_VIDEO = 9, ///< Has a file video event.
QAV_EVENT_ID_ENDPOINT_NO_MEDIA_FILE_VIDEO = 10, ///< No file video event.
};
```

The "video data types" mentioned earlier can be defined in the event.
You only need to configure the listener, listen for the event QAV_EVENT_ID_ENDPOINT_HAS_CAMERA_VIDEO which indicates that the camera is enabled in the proxy method, and add the user's rendering image to the interface.

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

```
// Audio/video event callback
- (BOOL)onEndpointsUpdateInfo:(QAVUpdateEvent)event updateList:(NSArray *)endpoints {
switch (event) {
case QAV_EVENT_ID_ENDPOINT_HAS_CAMERA_VIDEO:
{
/*
Create and add rendering views, pass userID and rendering image type. QAVVIDEO_SRC_TYPE_CAME
RA (camera image) is entered here.
*/
ILiveFrameDispatcher *frameDispatcher = [[ILiveRoomManager getInstance] getFrameDispatcher];
ILiveRenderView *renderView = [frameDispatcher addRenderAt:self.view.bounds foruserId:[endpoints.
firstObject userId] srcType:QAVVIDEO_SRC_TYPE_CAMERA];
[self.view addSubview:renderView];
}
break;
}
return YES;
}
```

If everything goes well, you can see the live image when you successfully create the room and are redirected to the LVB page.

> Since this document is intended to create a live room and obtain images, the processing of audio and video event callback is relatively simple, but this is also fundamental. More information on this method will be described in subsequent documents. You can listen for different events as needed. For example, you can listen for the events of members entering and exiting the room and display the notification of members entering and exiting the room on the interface.

The listening of the room disconnection event complies with the `ILiveRoomDisconnectListener` protocol and implements the following methods:

```
/**
Prompt when SDK actively exits the room. This callback method indicates that the SDK has actively ex
ited the room. The SDK will actively exit the room due to the 30s timeout of heartbeat packet. The Ap
p listens for this event of exiting a room and processes it accordingly.

@param reason Reasons for exiting the room. See the error code for specific values

@return YES Operation succeeded
```

```
*/
- (BOOL)onRoomDisconnect:(int)reason;
```

This method is a callback after the SDK actively exits the room, and developers can process it in methods based on the business needs.

Exit a room

Call the API of exiting a room of ILiveSDk.
Call the API in the dealloc method of the LVB controller.

> Note: Do not reference your LVB controller circularly. Otherwise, the dealloc method will not be called.

```
//Call the API for exiting a room when a room is terminated
- (void)dealloc {
[[ILiveRoomManager getInstance] quitRoom:^{
NSLog(@"Exited room successfully");
} failed:^(NSString *module, int errId, NSString *errMsg) {
NSLog(@"Failed to exit the room %d : %@", errId, errMsg);
}];
}
```

After a user exits the room, the resources in the room will be reclaimed, including roomID. You can create a new room with the same roomID.

FAQ

Failed to enter a room, and was prompted for required permission

Make sure that the room ticket field (privateMapKey) is configured correctly.

> The privateMapKey field is required for new users, and existing users (do not need room tickets) need to configure the field at initialization
>
> ```
> [[ILiveSDK getInstance] setChannelMode:E_ChannelIMSDK withHost:@""];
> ```

Callback for failed operations. Error code 1003 or 8011

1. Operations of entering/exiting rooms are linearly exclusive. If your request is too frequent, SDK may throw 8011, which means the last operation must be completed (callback and return) before continuing (to enter/exit the room).
2. You can only enter one room at a time. If you do not exit the last room before creating (or entering) a new room, 1003 is thrown. In this case, you must exit the last room first.

Email

If you have any questions, send us an email to trtcfb@qq.com.

PC

Last updated：2018-09-28 17:07:47

This document describes how to create a room in the client, and enable the camera and microphone to see your own video images.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Concepts

- Room
- privateMapKey
- Role configuration
- Video rendering
  The captured video data must be rendered. This process is called video rendering.

Setting a callback for video data.

To obtain video data, you must set callback functions for local and remote video data before creating/entering the room.

```
//Callback for local video data
void OnLocalVideo(const LiveVideoFrame* video_frame, void* data)
{
}

//Callback for remote video data
void OnRemoteVideo(const LiveVideoFrame* video_frame, void* data)
{
}

GetILive()->setLocalVideoCallBack(OnLocalVideo, NULL);
GetILive()->setRemoteVideoCallBack(OnRemoteVideo, NULL);
```

Creating a Room

To create a room, you need to enter the iLiveRoomOption structure to describe the information of the created room. The sample code is as follows:

```
//Notification of status changes of members in the room (such as enabling/disabling cameras)
void OnMemStatusChange(E_EndpointEventId eventId, const Vector<String> &ids, void* data)
{
}

iLiveRoomOption roomOption;
roomOption.privateMapKey = privateMapKey; // Configure a room ticket
roomOption.roomId = RoomId; //ID of the room to be created
roomOption.authBits = AUTH_BITS_DEFAULT; //With all permissions
roomOption.controlRole = "LiveMaster"; //Use the "LiveMaster" role configured on Spear
roomOption.memberStatusListener = OnMemStatusChange;//Callback indicating status change of m
embers in the room
roomOption.data = NULL;//void* data pointer returned intact in callback;

GetILive()->createRoom(roomOption, [](void* data) {
//Room created successfully
}, [](const int code, const char *desc, void* data) {
//Failed to create a room
}, NULL);
```

Enabling Camera and Microphone

After you create a room, you don't need to call an API to enter the room. The camera and microphone can be enabled to perform audio/video data upstream.

```
//Enable the camera
Vector< Pair<String/*id*/, String/*name*/> > cameraList;
GetILive()->getCameraList(cameraList);//Obtain a list of available cameras
if (cameraList.size() > 0 )
{
GetILive()->openCamera(cameraList[0].first); //Enable the first camera (default camera)
}

//Enable the microphone
Vector< Pair<String/*id*/, String/*name*/> > micList;
GetILive()->getMicList(micList);//Obtain a list of available microphones
```

```
if (micList.size() > 0)
{
GetILive()->openMic(micList[0].first); //Enable the first microphone (default microphone)
}
```

In this case, the OnLocalVideo callback function configured previously receives every frame of video data. The data size of each video frame can be printed in the callback to verify whether the video data is received.

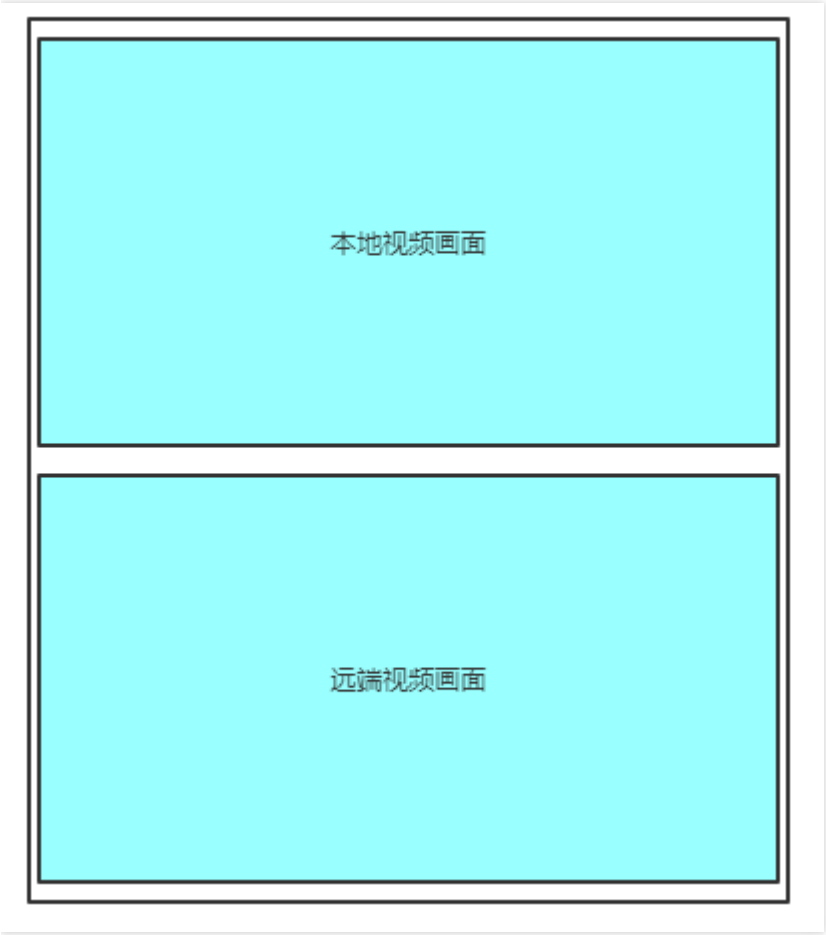
```
void OnLocalVideo(const LiveVideoFrame* video_frame, void* data)
{
printf("frame size: %d\n", video_frame->dataSize);
}
```

Video Rendering

To see the video image, the received video image from the local camera must be rendered. The iLiveSDK provides a rendering module (iLiveRootView) where a window handle is assigned and a View (user information of the image to be rendered) is added to the rendering module before transmitting each frame of video image to the rendering module for rendering. You can read Rendering Module Usage Documentation carefully and understand it with the demo source code in this example.

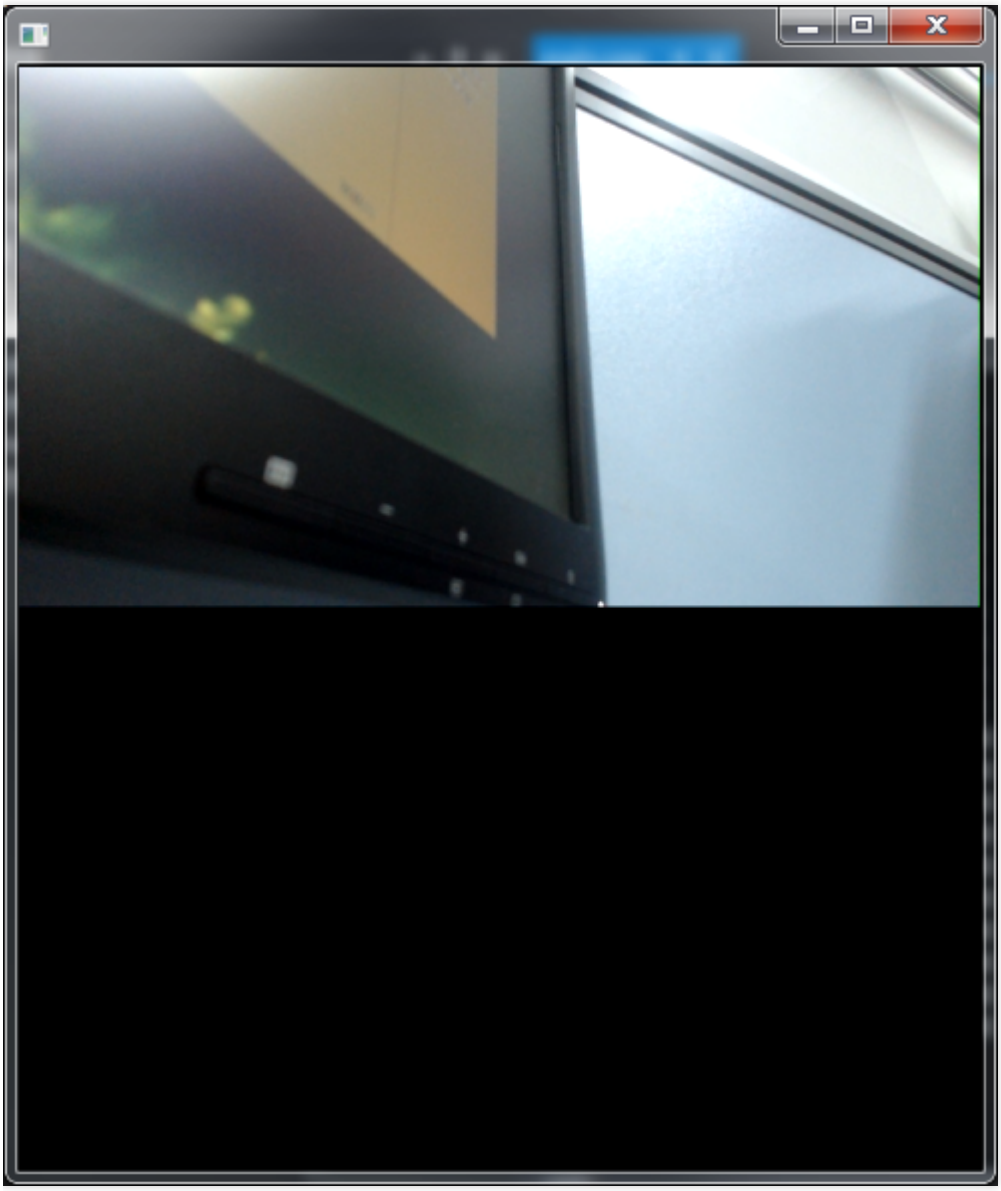
Interface Design

In this demo, a window is created for the follow-up courses, with the upper part displaying local video images and the lower part displaying remote video images, as shown below:



To reduce the complexity, this demo only demonstrates the audio/video communication between two people. In actual development, you can add multiple rendering views as needed.

Execution Results



FAQ

Failed to enter a room, and was prompted for required permission

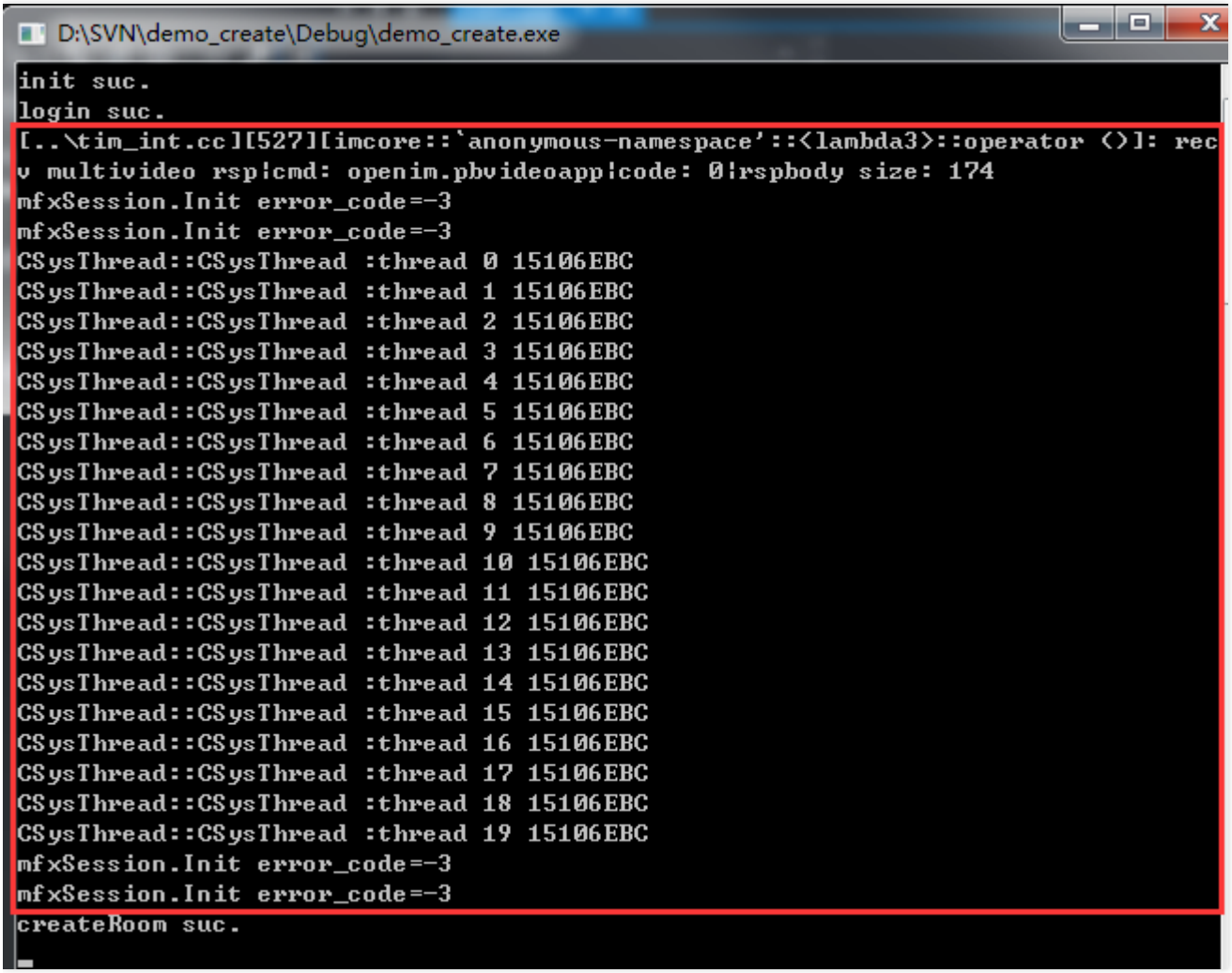
Make sure that the room ticket field (privateMapKey) is configured correctly.

The privateMapKey field is required for new users, and existing users (do not need room tickets) need to configure the field at initialization

```
GetILive()->setChannelMode(E_ChannelIMSDK);
```

Useless information is outputted on the console:

When some APIs of the SDK are called, useless information may be outputted on the console. For example, the output results after creating a room are shown as follows:



This is because iLiveSDK uses other SDK internally. It actually shows the printing output information of other SDKs, which will not affect the actual usage and can be ignored.

Email

If you have any questions, send us an email to trtcfb@qq.com.

Mac

Last updated：2018-09-28 17:06:55

This document describes how to create a room, obtain LVB video images and exit the room when appropriate.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Concepts

- Room
- privateMapKey
- Role configuration
- Rendering control
  You need a platform to display the obtained video data. That is the rendering control.
- Video data types
  Tencent Cloud supports performing the upstream of one mainstream and one substream simultaneously in one account. Videos from different sources of video streams are classified into the following types: camera, screen sharing, and video playing (generated from PC).

Video Type	Stream Type	Description
Camera	Mainstream	Data collected from the camera
Screen sharing	Substream	Generated from screen sharing
Video playing	Substream	Generated from video file playing

Procedure

Creating a room

A room can be created by `ILiveRoomManager.h`, which requires two key parameters, room ID (roomId) and room configuration object (option):

- The roomId can be entered by users in an interface (see the demo), or hard-coded in codes for testing. The roomId must conform to the rules described in Preliminary Information.

- The room configuration object must be created by users. The ILiveRoomOption is a class used to configure the instant messaging and other features of the audios and videos in the room to be created. You can use defaultHostLiveOption by default.

The result of creating a room is returned by calling back Block. You can process it in successful/failed callback based on your business logic.

```
> TCLiveRoomWC.m
// Import the header file
#import <ILiveSDK/ILiveCoreHeader.h>

// Create a room
- (void)enterRoom{
//Step 3: Configure a room
ILiveRoomOption *option = [ILiveRoomOption defaultHostLiveOption];
option.imOption.imSupport = YES;
// Set audio/video listening in the room
option.memberStatusListener = self;
// Set room disconnection event listening
option.roomDisconnectListener = self;
option.firstFrameListener = self;
// This parameter indicates the audio/video specification used after a user enters a room. The value o
f the parameter is the role name configured by the customer in **Screen Setting** on the Tencent Clo
ud TRTC console (for example, the default role name is User, and you can set controlRole = @"user")
option.controlRole = self.role;
// Configure a room ticket
option.privateMapKey = privateMapKey;

//Step 4: Call the API to create a room and pass the room ID and room configuration object
[[ILiveRoomManager getInstance] createRoom:[self.roomID intValue] option:option succ:^{
NSLog(@"-----> create room succ");
} failed:^(NSString *module, int errId, NSString *errMsg) {
NSLog(@"-----> create room fail,%@ %d %@",module, errId, errMsg);
}];

}
```

After you enter the room, the camera and microphone are automatically enabled and the stream is automatically pushed, which can be set in the `ILiveRoomOption` configuration object passed when you create the room). In the live room, all audio/video events are notified to the listener through audio/video event callback. First, you must set a listener option.memberStatusListener = self;.

Listening events in the room

When creating a room, you can set room audio/video event listening and room disconnection event listening in the configuration object to listen for events in the room.
The listener of audio/video events complies with the `ILiveMemStatusListener` protocol and implements the following methods:

```
@protocol ILiveMemStatusListener <NSObject>
/**
The function of the notification of status changes of members in the room, through which the business side will be notified when the status members in the room change (such as publishing audios or videos).

@param event Status change ID. For more information, see the definition of QAVUpdateEvent
@param endpoints List of member IDs whose statuses change.

@return YES Operation succeeded
*/
- (BOOL)onEndpointsUpdateInfo:(QAVUpdateEvent)event updateList:(NSArray *)endpoints;
@end
```

Notes:
This method is a callback method for audio/video events in the room. If someone in the room enables the camera and microphone, the underlying SDK will call back this method to notify the listener. This method has two parameters, event and endpoints.

- The event is an enumerated value for events, which defines the types of events that could occur to members in the room, including entering and exiting the room, and enabling/disabling cameras and microphones.
- The endpoints is an array of QAVEndpoint objects, which specifies each user sending an event. This method is used to send notifications of specific types of changes in audios and videos and the users who make those changes. When an event change is listened for, some adjustments must be made on the interface. For example, when it finds that a user camera is enabled, a rendering image should be added to the interface to render the user's image.

The event has the following values:

```
typedef NS_ENUM(NSInteger, QAVUpdateEvent) {
QAV_EVENT_ID_NONE = 0, ///< Default value. Ignored..
QAV_EVENT_ID_ENDPOINT_ENTER = 1, ///< Event of entering a room..
QAV_EVENT_ID_ENDPOINT_EXIT = 2, ///< Event of exiting a room..
QAV_EVENT_ID_ENDPOINT_HAS_CAMERA_VIDEO = 3, ///< Has a camera video event.
QAV_EVENT_ID_ENDPOINT_NO_CAMERA_VIDEO = 4, ///< No camera video event.
QAV_EVENT_ID_ENDPOINT_HAS_AUDIO = 5, ///< Has an audio event.
QAV_EVENT_ID_ENDPOINT_NO_AUDIO = 6, ///< No audio event.
QAV_EVENT_ID_ENDPOINT_HAS_SCREEN_VIDEO = 7, ///< Has a screen video event.
QAV_EVENT_ID_ENDPOINT_NO_SCREEN_VIDEO = 8, ///< No screen video event.
QAV_EVENT_ID_ENDPOINT_HAS_MEDIA_FILE_VIDEO = 9, ///< Has a file video event.
QAV_EVENT_ID_ENDPOINT_NO_MEDIA_FILE_VIDEO = 10, ///< No file video event.
};
```

The "video data types" mentioned earlier can be defined in the event.
You only need to configure the listener, listen for the event QAV_EVENT_ID_ENDPOINT_HAS_CAMERA_VIDEO which indicates that the camera is enabled in the proxy method, and add the user's rendering image to the interface.

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

// Audio/video event callback
- (BOOL)onEndpointsUpdateInfo:(QAVUpdateEvent)event updateList:(NSArray *)endpoints {
switch (event) {
case QAV_EVENT_ID_ENDPOINT_HAS_CAMERA_VIDEO:
{
/*
Create and add rendering views, pass userID and rendering image type. QAVVIDEO_SRC_TYPE_CAME
RA (camera image) is entered here.
*/
ILiveFrameDispatcher *frameDispatcher = [[ILiveRoomManager getInstance] getFrameDispatcher];
ILiveRenderViewForMac *renderView = [frameDispatcher addRenderAt:CGRectMake(0, 0, self.videoLa
youtView.frame.size.width, self.videoLayoutView.frame.size.height - 20) forIdentifier:endoption.identif
ier srcType:QAVVIDEO_SRC_TYPE_CAMERA];
renderView.identifier = endoption.identifier;
[self.window.contentView addSubview:renderView];
}
break;
}
```

```
return YES;
}
```

If everything goes well, you can see the live image when you successfully create the room and are redirected to the LVB page.

> Since this document is intended to create a live room and obtain images, the processing of audio and video event callback is relatively simple, but this is also fundamental. More information on this method will be described in subsequent documents. You can listen for different events as needed. For example, you can listen for the events of members entering and exiting the room and display the notification of members entering and exiting the room on the interface.

The listening of the room disconnection event complies with the `ILiveRoomDisconnectListener` protocol and implements the following methods:

```
/**
Prompt when SDK actively exits the room. This callback method indicates that the SDK has actively ex
ited the room. The SDK will actively exit the room due to the 30s timeout of heartbeat packet. The Ap
p listens for this event of exiting a room and processes it accordingly.

@param reason Reasons for exiting the room. See the error code for specific values

@return YES Operation succeeded
*/
- (BOOL)onRoomDisconnect:(int)reason;
```

This method is a callback after the SDK actively exits the room, and developers can process it in methods based on the business needs.

Exit a room

Call the API of exiting a room of ILiveSDk.
Call the API when the window is closed.

```
//Close the window and exit the room
-(void)windowWillClose:(NSNotification *)notification{
[[ILiveRoomManager getInstance] quitRoom:^{
NSLog(@"-----> quit room succ");
} failed:^(NSString *module, int errId, NSString *errMsg) {
NSLog(@"-----> quit room fail,%@ %d %@",module, errId, errMsg);
```

```
}];
}
```

After a user exits the room, the resources in the room will be reclaimed, including roomID. You can create a new room with the same roomID.

FAQ

Failed to enter a room, and was prompted for required permission

Make sure that the room ticket field (privateMapKey) is configured correctly.

> The privateMapKey field is required for new users, and existing users (do not need room tickets) need to configure the field at initialization
>
> ```
> [[ILiveSDK getInstance] setChannelMode:E_ChannelIMSDK withHost:@""];
> ```

Callback for failed operations. Error code 1003 or 8011

1. Operations of entering/exiting rooms are linearly exclusive. If your request is too frequent, SDK may throw 8011, which means the last operation must be completed (callback and return) before continuing (to enter/exit the room).
2. You can only enter one room at a time. If you do not exit the last room before creating (or entering) a new room, 1003 is thrown. In this case, you must exit the last room first.

Email

If you have any questions, send us an email to trtcfb@qq.com.

Enter a Room
Android

Last updated：2018-09-28 17:15:02

This document describes how a viewer enters a room and enables the camera and the microphone to interact with other users.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Concepts

- TRTC application
- privateMapKey
- Role configuration
- Camera ID (cameraId)
  Android phones usually have two cameras: a front camera and a rear camera. The SDK can distinguish between them by cameraId.

Constant	Description
ILiveConstants.NONE_CAMERA	Invalid camera ID (Generally indicating that the camera is not enabled)
ILiveConstants.FRONT_CAMERA	Front camera ID
ILiveConstants.BACK_CAMERA	Rear camera ID

Entering a Room

The room module in entering a room is basically the same as that in Creating a room, except that the method used here is joinRoom.

```
// Enter a room
public int joinRoom(int roomId){
ILiveRoomOption option = new ILiveRoomOption()
.privateMapKey(privateMapKey) // Room ticket
.imsupport(false) // Do not need the IM feature
.exceptionListener(this) // Listen on exceptional events
.roomDisconnectListener(this) // Listen on room disconnection events
.controlRole("user") // Use the user role
.autoCamera(false) // Do not enable the camera when the user enters a room
.autoMic(false); // Do not enable the microphone when the user enters a room

return ILiveRoomManager.getInstance().joinRoom(roomId, option, new ILiveCallBack() {
@Override
public void onSuccess(Object data) {
roomView.onEnterRoom();
}

@Override
public void onError(String module, int errCode, String errMsg) {
roomView.onEnterRoomFailed(module, errCode, errMsg);
}
});
}
```

Enabling Capturing Devices (Camera and Microphone)

If interaction with other users via upstream audio/video is required, you need to add two APIs to the room module to control the camera and the microphone respectively.

```
// Camera
public int enableCamera(int cameraId, boolean enable){
return ILiveRoomManager.getInstance().enableCamera(cameraId, enable);
}
// Microphone
public int enableMic(boolean enable){
return ILiveRoomManager.getInstance().enableMic(enable);
}
```

UI Development

We can enrich the interface by placing a set of buttons on the top of the rendering control for switching roles, and enabling or disabling the camera and the microphone.
We won't go into details here.

FAQ

Failed to enter a room, and was prompted for required permission

Make sure that the room ticket field (privateMapKey) is configured correctly.

> The privateMapKey field is required for new users, and existing users (do not need room tickets) need to configure the field at initialization
>
> ```
> ILiveSDK.getInstance().setChannelMode(CommonConstants.E_ChannelMode.E_ChannelIMSDK);
> ```

Email

If you have any questions, send us an email to trtcfb@qq.com.

iOS

Last updated：2018-09-28 17:15:47

This document describes how a viewer enters a room and enables the camera and the microphone to interact with other users.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Entering a Room

The method for entering a room is in ILiveRoomManager.h, and is the same as that for creating a room. This method also requires passing in two parameters, room ID (roomId) and room configuration object (option) (the roomId of the room to be entered must be the same as the created roomId). When creating a configuration object, we should disable the auto start of the camera and the microphone.

```
- (IBAction)onJoinRoom:(id)sender {
// 1. Create a live room page
LiveRoomViewController *liveRoomVC = [[LiveRoomViewController alloc] init];

// 2. Create a room configuration object
ILiveRoomOption *option = [ILiveRoomOption defaultHostLiveOption];
// Configure a room ticket
option.privateMapKey = privateMapKey;
option.imOption.imSupport = NO;
// Do not enable the camera automatically
option.avOption.autoCamera = NO;
// Do not enable the microphone automatically
option.avOption.autoMic = NO;
// Set audio/video listening in the room
option.memberStatusListener = liveRoomVC;
// Set room disconnection event listening
option.roomDisconnectListener = liveRoomVC;

// This parameter indicates the audio/video specification used after a user enters a room. The value o
f the parameter is the role name configured by the customer in Screen Setting on the Tencent Cloud
TRTC console (for example, the default role name is user, and you can set controlRole = @"user")
```

```
option.controlRole = #The role name configured on the Tencent Cloud console#;

// 3. Call the API for creating a room with roomId and option passed
[[ILiveRoomManager getInstance] joinRoom:[self.roomIDTF.text intValue] option:option succ:^{
// Entered the room successfully. The user will be redirected to the room page
[self.navigationController pushViewController:liveRoomVC animated:YES];animated:YES];
} failed:^(NSString *module, int errId, NSString *errMsg) {
// Failed to enter the room
NSLog(@"Failed to enter the room errId:%d errMsg:%@",errId, errMsg);
}];
}
```

After successfully entering a room, the user will be directly redirected to the room page.

Video Interaction

To start video interactions with other users in the room, enable the camera and the microphone:

```
/**
Enable/disable the camera

@param cameraPos Camera position cameraPos
@param bEnable YES: Enable NO: Disable
@param succ Callback is successful
@param fail Callback failed
*/
- (void)enableCamera:(cameraPos)cameraPos enable:(BOOL)bEnable succ:(TCIVoidBlock)succ failed:
(TCIErrorBlock)fail;

/**
Enable/disable the microphone

@param bEnable YES: Enable NO: Disable
@param succ Callback is successful
@param fail Callback failed
*/
- (void)enableMic:(BOOL)bEnable succ:(TCIVoidBlock)succ failed:(TCIErrorBlock)fail;
```

After triggering an action, call the corresponding API.

Audio/Video Event Listening

The audio/video event callback method handles events involving multiple users, mainly the camera enabling/disabling events.

> If QAV_EVENT_ID_ENDPOINT_HAS_CAMERA_VIDEO is detected, which indicates that a user enables the camera, add the rendered view of the user.
> If QAV_EVENT_ID_ENDPOINT_NO_CAMERA_VIDEO is detected, which indicates that a user disables the camera, remove the rendered view of the user.

The frame of a rendered view is not fixed and needs to be calculated according to the current number of rendered views and product demands. In the Demo provided in this document, we demonstrate a simple top-down equal-division layout for rendered views. You can also define your own layout logic according to your needs.

```
// Audio/video event callback
- (BOOL)onEndpointsUpdateInfo:(QAVUpdateEvent)event updateList:(NSArray *)endpoints {
if (endpoints.count <= 0) {
return NO;
}
for (QAVEndpoint *endpoint in endpoints) {
switch (event) {
case QAV_EVENT_ID_ENDPOINT_HAS_CAMERA_VIDEO:
{
/*
Create and add rendering views, pass userID and rendering image type. QAVVIDEO_SRC_TYPE_CAME
RA (camera image) is entered here.
*/
ILiveFrameDispatcher *frameDispatcher = [[ILiveRoomManager getInstance] getFrameDispatcher];
ILiveRenderView *renderView = [frameDispatcher addRenderAt:CGRectZero foruserId:endpoint.userId
srcType:QAVVIDEO_SRC_TYPE_CAMERA];
[self.view addSubview:renderView];
[self.view sendSubviewToBack:renderView];
// The number of users enabling the camera in the room changes and the rendered views are laid ou
t again
[self onCameraNumChange];
}
break;
case QAV_EVENT_ID_ENDPOINT_NO_CAMERA_VIDEO:
{
// Remove a rendered view
ILiveFrameDispatcher *frameDispatcher = [[ILiveRoomManager getInstance] getFrameDispatcher];
ILiveRenderView *renderView = [frameDispatcher removeRenderViewFor:endpoint.userId srcType:QAV
```

```
VIDEO_SRC_TYPE_CAMERA];
[renderView removeFromSuperview];
// The number of users enabling the camera in the room changes and the rendered views are laid ou
t again
[self onCameraNumChange];
}
break;
default:
break;
}
}
return YES;
}

// This is called when the number of users enabling the camera in the room changes to re-layout all t
he rendered views. Here we simply arrange the rendered views equally from top to bottom.
- (void)onCameraNumChange {
// Obtain all rendered views
NSArray *allRenderViews = [[TILLiveManager getInstance] getAllAVRenderViews];

// Detect exceptions
if (allRenderViews.count == 0) {
return;
}

// Calculate and set the frame for each rendered view
CGFloat renderViewHeight = [UIScreen mainScreen].bounds.size.height / allRenderViews.count;
CGFloat renderViewWidth = [UIScreen mainScreen].bounds.size.width;
__block CGFloat renderViewY = 0.f;
CGFloat renderViewX = 0.f;

[allRenderViews enumerateObjectsUsingBlock:^(ILiveRenderView *renderView, NSUInteger idx, BOOL
* _Nonnull stop) {
renderViewY = renderViewY + renderViewHeight * idx;
CGRect frame = CGRectMake(renderViewX, renderViewY, renderViewWidth, renderViewHeight);
renderView.frame = frame;
}];
}
```

Note:

1. When adding rendered views, you don't need to worry about duplication. The SDK only allows adding one rendered view of the same type of video source for the same user.

2. The SDK only allows a maximum of 10 rendered views in it.

FAQ

Failed to enter a room, and was prompted for required permission

Make sure that the room ticket field (privateMapKey) is configured correctly.

The privateMapKey field is required for new users, and existing users (do not need room tickets) need to configure [[ILiveSDK getInstance] setChannelMode:E_ChannelIMRestAPI withHost:@""]; at initialization

```
[[ILiveSDK getInstance] setChannelMode:E_ChannelIMSDK withHost:@""];
```

Failed to change the role. Error code is -1.

This means that the configuration backend cannot find the role to be changed to. Check whether the role name is entered correctly (case-sensitive).

Email

If you have any questions, send us an email to trtcfb@qq.com.

PC

Last updated：2018-09-28 17:13:46

This document describes how a viewer enters a created room and interacts with other users.

Downloading Source Code

You can download the complete demo code used in this document.
Click to download

Entering a Room

Just like creating a room, you need to complete initialization and login before you can enter a room. You also need to enter the iLiveRoomOption structure to describe the information of the room you want to enter, and then call the API joinRoom() to enter the room.

```
//Notification of status changes of members in the room (such as enabling/disabling cameras)
void OnMemStatusChange(E_EndpointEventId eventId, const Vector<String> &ids, void* data)
{
}

iLiveRoomOption roomOption;
roomOption.privateMapKey = privateMapKey; // Configure a room ticket
roomOption.roomId = RoomId; //ID of the room to be entered
roomOption.authBits = AUTH_BITS_DEFAULT; //With all permissions
roomOption.controlRole = "user"; //Use the "user" role configured on Spear
roomOption.memberStatusListener = OnMemStatusChange;//Callback indicating change of member status
roomOption.data = NULL;//void* data pointer returned intact in callback.

GetILive()->joinRoom(roomOption, [](void* data) {
//Entered the room successfully
}, [](const int code, const char *desc, void* data) {
//Failed to enter the room
}, NULL);
```

Enabling Camera and Microphone

After entering a room, a user can enable the camera and the microphone to interact with other users in exactly the same way as that in Creating a Room.
For more information, see Creating a Room - Enabling Camera and Microphone.

Remote Video Rendering

After a user enter a room, the SDK will automatically request the video images of other members in the room, which means the remote video data can be obtained in this callback at this time. You need to render the remote screen in the same way as that in Creating a Room. For more information, see Creating a Room - Video Rendering.

In this way, members in the room can have audio/video interactions.

Source Code Description

- Test description
  The interoperability test between the Demo provided in this document and the complete Demo in Creating a Room should be conducted separately on two computers, because it is hard coded to enable the first camera and the first microphone. If you want to do the test on one computer, the computer must have at least two cameras and microphones, and two users cannot use the same device.

Execution Results

FAQ

Failed to enter a room, and was prompted for required permission

Make sure that the room ticket field (privateMapKey) is configured correctly.

The privateMapKey field is required for new users, and existing users (do not need room tickets) need to configure [[ILiveSDK getInstance] setChannelMode:E_ChannelIMRestAPI withHost:@""]; at initialization

```
GetILive()->setChannelMode(E_ChannelIMSDK);
```

Email

If you have any questions, send us an email to trtcfb@qq.com.

Mac

Last updated：2018-09-28 17:11:57

This document describes how a viewer enters a room and enables the camera and the microphone to interact with other users.

Downloading Source Code

You can download the complete demo code used in this document.
Download Demo Code

Entering a Room

The method for entering a room is in ILiveRoomManager.h, and is the same as that for creating a room. This method also requires passing in two parameters, room ID (roomId) and room configuration object (option) (the roomId of the room to be entered must be the same as the created roomId). When creating a configuration object, we should disable the auto start of the camera and the microphone.

```
- (IBAction)onJoinRoom:(id)sender {

// Create a room configuration object
ILiveRoomOption *option = [ILiveRoomOption defaultHostLiveOption];
// Configure a room ticket
option.privateMapKey = privateMapKey;
option.imOption.imSupport = NO;
// Do not enable the camera automatically
option.avOption.autoCamera = NO;
// Do not enable the microphone automatically
option.avOption.autoMic = NO;
// Set audio/video listening in the room
option.memberStatusListener = liveRoomVC;
// Set room disconnection event listening
option.roomDisconnectListener = liveRoomVC;

// This parameter indicates the audio/video specification used after a user enters a room. The value o
f the parameter is the role name configured by the customer in Screen Setting on the Tencent Cloud
TRTC console (for example, the default role name is user, and you can set controlRole = @"user")
option.controlRole = #The role name configured on the Tencent Cloud console#;
```

```
// Call the API for entering a room with roomId and option passed
[[ILiveRoomManager getInstance] joinRoom:[self.roomID intValue] option:option succ:^{
// Entered the room successfully
NSLog(@"-----> join room succ");
} failed:^(NSString *module, int errId, NSString *errMsg) {
// Failed to enter the room
NSLog(@"Failed to enter the roomerrId:%d errMsg:%@",errId, errMsg);
}];
}
```

After successfully entering a room, the user will be directly redirected to the room page.

Video Interaction

To start video interactions with other users in the room, enable the camera and the microphone:

```
/**
Enable/disable the camera

@param cameraPos Camera position cameraPos
@param bEnable YES: Enable NO: Disable
@param succ Callback is successful
@param fail Callback failed
*/
- (void)enableCamera:(cameraPos)cameraPos enable:(BOOL)bEnable succ:(TCIVoidBlock)succ failed:
(TCIErrorBlock)fail;

/**
Enable/disable the microphone

@param bEnable YES: Enable NO: Disable
@param succ Callback is successful
@param fail Callback failed
*/
- (void)enableMic:(BOOL)bEnable succ:(TCIVoidBlock)succ failed:(TCIErrorBlock)fail;
```

After triggering an action, call the corresponding API.

Audio/Video Event Listening

The audio/video event callback method handles events involving multiple users, mainly the camera enabling/disabling events.

If QAV_EVENT_ID_ENDPOINT_HAS_CAMERA_VIDEO is detected, which indicates that a user enables the camera, add the rendered view of the user.
If QAV_EVENT_ID_ENDPOINT_NO_CAMERA_VIDEO is detected, which indicates that a user disables the camera, remove the rendered view of the user.

The frame of a rendered view is not fixed and needs to be calculated according to the current number of rendered views and product demands. In the Demo provided in this document, we demonstrate a layout that supports simultaneous display of 4 video streams. We place the primary screen of the rendered view on the lower layer, and the other 3 secondary screens horizontally above the primary screen. You can also define your own layout logic according to your needs.

```
// Audio/video event callback
- (BOOL)onEndpointsUpdateInfo:(QAVUpdateEvent)event updateList:(NSArray *)endpoints {
if (endpoints.count <= 0) {
return NO;
}
for (QAVEndpoint *endpoint in endpoints) {
switch (event) {
case QAV_EVENT_ID_ENDPOINT_HAS_CAMERA_VIDEO:
{
/*
Create and add rendering views, pass userID and rendering image type. QAVVIDEO_SRC_TYPE_CAME
RA (camera image) is entered here.
*/
ILiveFrameDispatcher *frameDispatcher = [[ILiveRoomManager getInstance] getFrameDispatcher];
ILiveRenderViewForMac *renderView = [frameDispatcher addRenderAt:CGRectZero forIdentifier:endo
ption.identifier srcType:QAVVIDEO_SRC_TYPE_CAMERA];
renderView.identifier = endoption.identifier;
// The number of users enabling the camera in the room changes and the rendered views are laid ou
t again
[self updateVideoFrame:renderView];
}
break;
case QAV_EVENT_ID_ENDPOINT_NO_CAMERA_VIDEO:
{
// Remove a rendered view
ILiveFrameDispatcher *frameDispatcher = [[ILiveRoomManager getInstance] getFrameDispatcher];
ILiveRenderViewForMac *renderView = [frameDispatcher removeRenderViewFor:endoption.identifier
srcType:QAVVIDEO_SRC_TYPE_CAMERA];
[renderView removeFromSuperview];
```

```
// The number of users enabling the camera in the room changes and the rendered views are laid ou
t again
[self updateVideoFrame:nil];
}
break;
default:
break;
}
}
return YES;
}


// This is called when the number of users enabling the camera in the room changes to re-layout all t
he rendered views
- (void)updateVideoFrame:(ILiveRenderViewForMac *)renderView{
if (renderView && ![self.window.contentView.subviews containsObject:renderView]) {
[self.videoLayoutView addSubview:renderView];
}
NSArray *allRenderView = [[[ILiveRoomManager getInstance] getFrameDispatcher] getAllRenderView
s];

if (allRenderView.count == 1) {
ILiveRenderViewForMac *bigView = allRenderView[0];
bigView.frame = CGRectMake(0, 0, self.videoLayoutView.frame.size.width, self.videoLayoutView.fram
e.size.height - 20);
[bigView viewWithTag:1001].frame = CGRectMake(bigView.frame.size.width/2, bigView.frame.size.hei
ght - 10, 300, 20);
}
else if (allRenderView.count == 2){
ILiveRenderViewForMac *bigView = allRenderView[0];
ILiveRenderViewForMac *smallView1 = allRenderView[1];
bigView.frame = CGRectMake(0, 0, self.videoLayoutView.frame.size.width, self.videoLayoutView.fram
e.size.height - 160 );
smallView1.frame = CGRectMake(self.videoLayoutView.frame.size.width/2 - 90, self.window.frame.siz
e.height - 150, 180, 120);

}
else if (allRenderView.count == 3){
ILiveRenderViewForMac *bigView = allRenderView[0];
ILiveRenderViewForMac *smallView1 = allRenderView[1];
ILiveRenderViewForMac *smallView2 = allRenderView[2];
bigView.frame = CGRectMake(0, 0, self.videoLayoutView.frame.size.width, self.videoLayoutView.fram
e.size.height - 160 );
smallView1.frame = CGRectMake(self.videoLayoutView.frame.size.width/2 - 180 - 10, self.window.fra
me.size.height - 150, 180, 120);
```

```
smallView2.frame = CGRectMake(self.videoLayoutView.frame.size.width/2 + 10, self.window.frame.siz
e.height - 150, 180, 120);
}
else if (allRenderView.count == 4 || allRenderView.count > 4 ){
ILiveRenderViewForMac *bigView = allRenderView[0];
ILiveRenderViewForMac *smallView1 = allRenderView[1];
ILiveRenderViewForMac *smallView2 = allRenderView[2];
ILiveRenderViewForMac *smallView3 = allRenderView[3];
bigView.frame = CGRectMake(0, 0, self.videoLayoutView.frame.size.width, self.videoLayoutView.fram
e.size.height - 160 );
smallView1.frame = CGRectMake(self.videoLayoutView.frame.size.width/2 - 180 - 90 - 10, self.windo
w.frame.size.height - 150, 180, 120);
smallView2.frame = CGRectMake(self.videoLayoutView.frame.size.width/2 - 90, self.window.frame.siz
e.height - 150, 180, 120);
smallView3.frame = CGRectMake(self.videoLayoutView.frame.size.width/2 + 90 + 10, self.window.fra
me.size.height - 150, 180, 120);
}
}
```

Note:

1. When adding rendered views, you don't need to worry about duplication. The SDK only allows adding one rendered view of the same type of video source for the same user.
2. The SDK only allows a maximum of 10 rendered views in it.

FAQ

Failed to enter a room, and was prompted for required permission?

Make sure that the room ticket field (privateMapKey) is configured correctly.
The privateMapKey field is required for new users, and existing users (do not need room tickets) need to configure [[ILiveSDK getInstance] setChannelMode:E_ChannelIMRestAPI withHost:@""]; at initialization

```
[[ILiveSDK getInstance] setChannelMode:E_ChannelIMSDK withHost:@""];
```

Failed to change the role. Error code is -1.

This means that the configuration backend cannot find the role to be changed to. Check whether the role name is entered correctly (case-sensitive).

Email

If you have any questions, send us an email to trtcfb@qq.com.