eSDK ICP V200R001C10 Development Guide 01 (CC, iOS)

Issue 01

Date 2017-03-22

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions

HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.huawei.com

Email: support@huawei.com

Contents

1 What Is eSDK ICP

What Is ICP

Huawei Integrated Communication Platform (ICP) can interwork with service systems, including the firefighting, police, first aid, video surveillance, wireless communications, public telephone, and computer-assisted dispatch (CAD) systems through multiple interfaces. The ICP is used to report urgent incidents or seek help in case of emergencies. Featuring unified alarm receiving, unified command, and joint action, the ICP provides services for citizens upon emergencies, ensuring public security. The ICP has enhanced the cooperation between police units so that they can respond to special, urgent, and critical incidents effectively and efficiently.

As the core of the converged command system, the ICP builds a connection among multiple voice networks, voice systems with multiple terminals, and various video systems, so that different devices, such as the fixed-line phones, mobile phones, trunking terminals, and telepresence endpoints can communicate with each other. Voice communication using the convergence of multiple networks enables unified command and quick distribution of information.

What Is CC SDK

As a sub-module of the eSDK ICP solution, CC SDK provides the call and call device control functions.

The SDK package provided by Huawei contains the following contents:

- **CC SDK package**

  SDK package for secondary development, including the software package and interface reference document.

- **Sample codes**

  Huawei SDK provides a series of sample codes for demonstrating how to invoke various interfaces, helping you to finish development of eSDK CC iOS interface-related services. For details, see the 3.2 Sample Code.

2 Development Guide Overview

This document provides guidance for developers to install and configure the eSDK CC iOS environment, invoke the eSDK CC iOS standard interfaces, and obtain technical support provided by the Huawei eSDK. This document consists of the following parts:

1. 3 Related Resources: Software, document resource website links, and technical support that may be involved in secondary development, including how to obtain materials from Huawei Developer Zone, download link of sample code, and how to apply for a remote lab.
2. 4 Hello World: Quick start guide to run the SDK. You should first read this chapter to learn how to download and install the SDK and configure the development environment.
3. 5 Typical Development Scenarios: Typical development scenarios of the eSDK CC iOS, consisting of development process, sample code, and precautions.
4. 6 Fault Locating Guide: Methods of locating common development problems.

Reading Guidance

- For a quick start, read 4 Hello World.
- To thoroughly understand secondary development of the core eSDK CC iOS services, read 5 Typical Development Scenarios.
- If you encounter any problems when you use the SDK, refer to 6 Fault Locating Guide or 3.7 Technical Support Channel.

3 Related Resources

3.1 Huawei Developer Zone

Visit the Cloud EC Section of Huawei Developer Zone to experience eSDK ICP service functions or obtain SDK tool packages or technical support for eSDK ICP secondary development.

3.2 Sample Code

You are advised to use Xcode to compile or execute the sample code.

The following provides the sample code list.

Demo	Description
eSDK_ICP_Demo_V2.1.10_iOS.zip	Typical scenario demo developed based on the eSDK CC iOS Sample, including the call and device control functions.

3.3 SDK Download Path

Visit the Resource Center of Huawei Developer Zone, choose **SDK** > **Cloud EC** > **ICP** > **eSDK CC iOS SDK**, and download the SDK software package of the required version.

The latest V2.1.10 version is recommended.

3.4 Interface Reference

The interface reference consists of the following contents:

Overview: mapping versions, usage background, scenarios, prerequisites, information that can be obtained, and functions to be performed.

Data types: detailed description of customized data types provided by the eSDK (including data structures, enumerations, and classes), including:

- Data type names
- Description of data structures that involve inheritance and nesting, such as structural body nesting relationships (a total nesting table including basis data types must be provided)
- Data members and their definitions

Interface description:

- Interface description: description of interface function, application scenarios, and usage.
- Usage description: precautions for using the interface function, usage limitation, interfaces with similar functions, associated interfaces, and prerequisites.
- Method definition: complete declaration of the interface function.
- Parameter description: description of parameter definition, value range, usage limitation, and relationships between parameters.
- Return value: return value of the interface function.
- Example: example that describes how to use the interface function. Key code is commented.

3.5 SDK Change History

The SDK is upgraded at intervals to support more services. You can visit the Huawei Developer Zone to view the change history, which includes the following information:

- SDK name
- Name of the mapping product
- Release time of the SDK version
- Current SDK version number
- Download link of the SDK demos and mapping documents
- Description of updated features

3.6 Free Application for the Remote Lab

Huawei eSDK Remote Lab Introduction

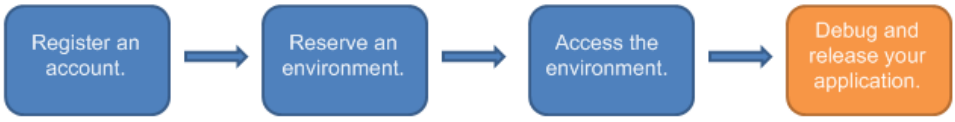
The Huawei remote lab provides 24/7 free cloud lab environment and real Huawei devices for developers to develop and commission applications online remotely. Using the remote lab self-management platform, developers can implement secondary development related to Huawei products without the need to purchase them and remotely test and authenticate their applications. Currently, Huawei remote lab has established 45 lab environments that are classified into 10 ecosystems, including cloud computing, SDN, big data, cloud EC, and enterprise mobile security.

For details, visit the remote lab homepage.

Advantages of the Remote Lab

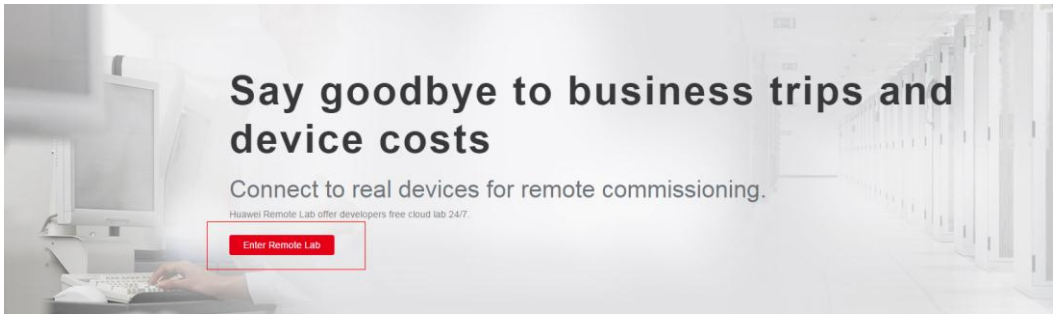
- Low entry barrier: Users who have been registered on the Huawei official website can apply to use the remote lab. Note that the environments that can be reserved, reservation duration, and the number of times the environments can be reserved are restricted.
- Hierarchical support: Environments are divided into different domains. Key developers and partners can access premium environments.
- High-speed connection between global resources: Huawei has established labs around the world with Suzhou remote lab as the center depending on the global high performance (delay less than 100 ms) backbone network and end-to-end support for applications.

How to Apply for the Remote Lab for Free

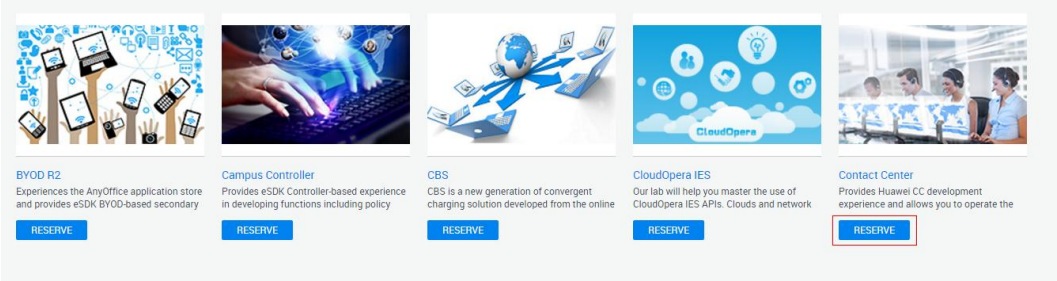


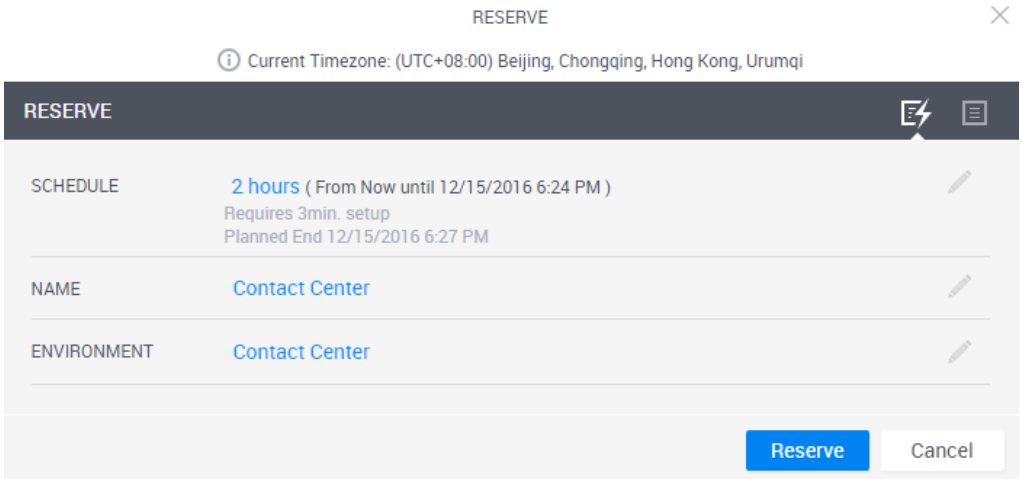
Step 1 Log in.

1. If you already register a Huawei account, use this account for login to the remote lab homepage.
2. If you have not registered on the Huawei official website, click http://developer.huawei.com/en/ict/remotelab to visit the Huawei remote lab website, click **Enter Remote Lab** on the remote lab homepage. On the registration page, enter the registration information. Then, log in to your registration's email account, open the confirmation email, and click the confirmation link to activate your Huawei account.



3. Reserve an environment.
   a. If you have successfully reserved an environment, and the environment is available, skip this step.
   b. When a Huawei account is successfully registered, the Huawei remote lab homepage is automatically displayed, and you are logged in. To use the contact center of the CC commissioning environment, click **RESERVE**, as shown in the following figure. The reservation duration is 2 hours by default.

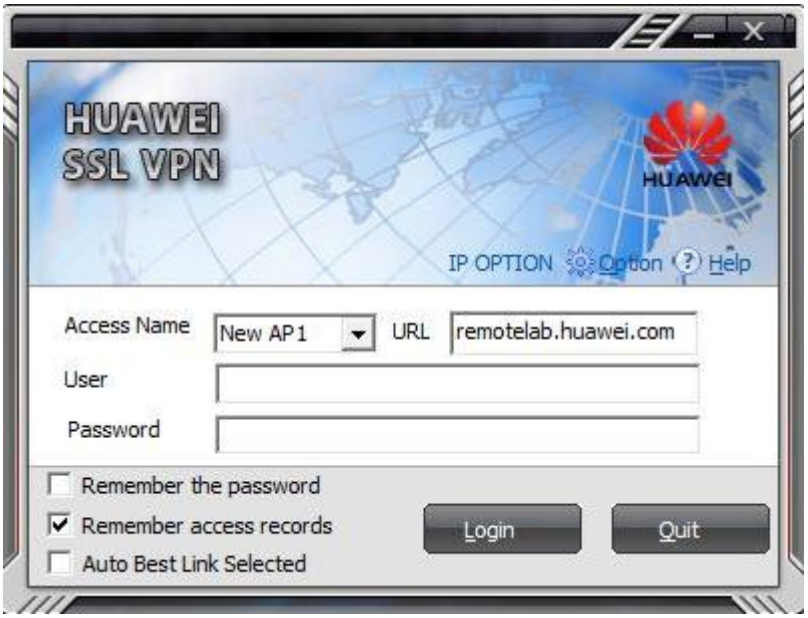




4. After you successfully reserve the environment, the system automatically sends the Secure Sockets Layer virtual private network (SVN) gateway address, user name, and password to your registration's email address. Log in to your registration's email account, open the email of the environment information, download the virtual private network over Secure Sockets Layer (SSL VPN) client as prompted, and install the client on the local PC. In subsequent steps, you need to use the environment information to log in to the SVN client and connect to the environment remotely.

Step 2 Access the environment.

1. If you have successfully accessed the Huawei remote lab, skip this step.
2. Open the SVN client. In the login window, enter the SVN gateway address, user name, and password you previously obtained and click **Login**.



Step 3 Commission and release the application.

Use the obtained CC platform account, password, IP address, and port information to log in, and commission your application. For details, see the CC platform login information in the remote lab operation guide.

----End

3.7 Technical Support Channel

If you have any problem when using the remote lab, contact us in one of the following ways:

- Huawei technical support hotline: 400-8828-000
- Huawei technical support email: esdk@huawei.com

4 Hello World

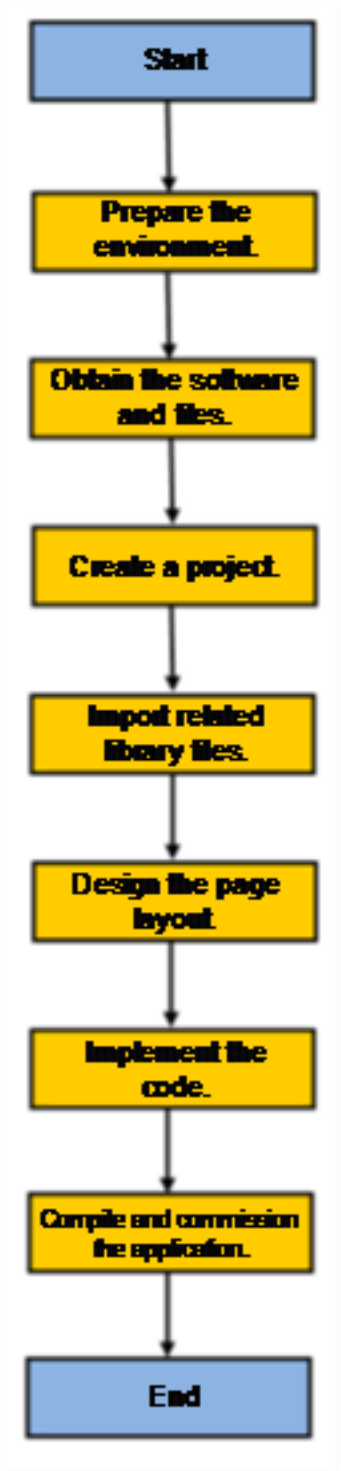
4.1 Overview

Hello World Development Process

The following example describes how to perform eSDK CC iOS secondary development in object-c.

For details about how to troubleshoot during development, see 6 Fault Locating Guide.

The following figure shows the Hello World demo development process.



4.2 Preparing the Environment

Development Tools

- Operating system: MAC OS 10.9 Mavericks or later
- XCode 8.0 or later

- iOS 8.0 or later

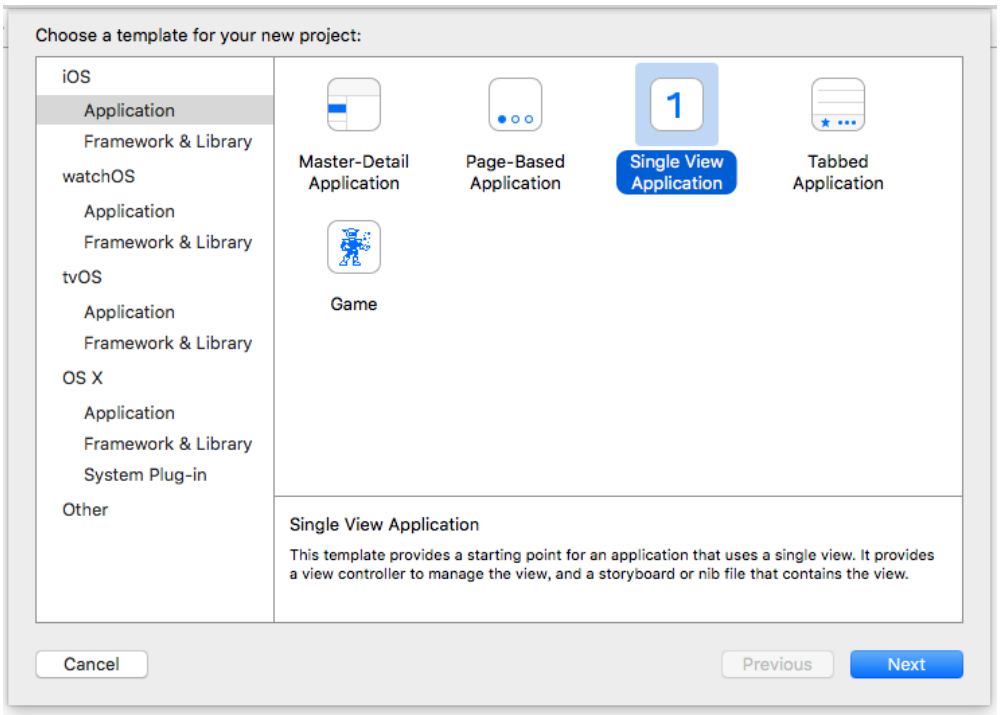
SDK Software Package

- SDK software package name: **eSDK_ICP_SDK_V2.1.10_iOS.zip**
- SDK software package download path: See 3.3 SDK Download Path.

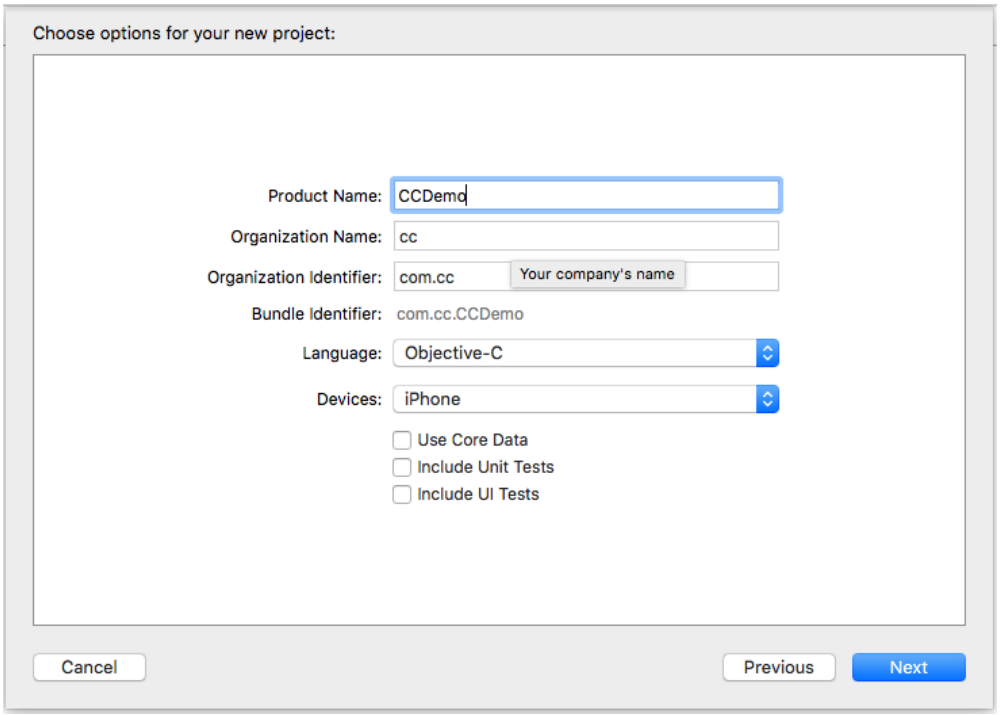
If the certificate verification function is required, the certificate needs to be placed in the **assets** folder of the project, for example, **assets/certs/server.cer**.

4.3 Creating a Project

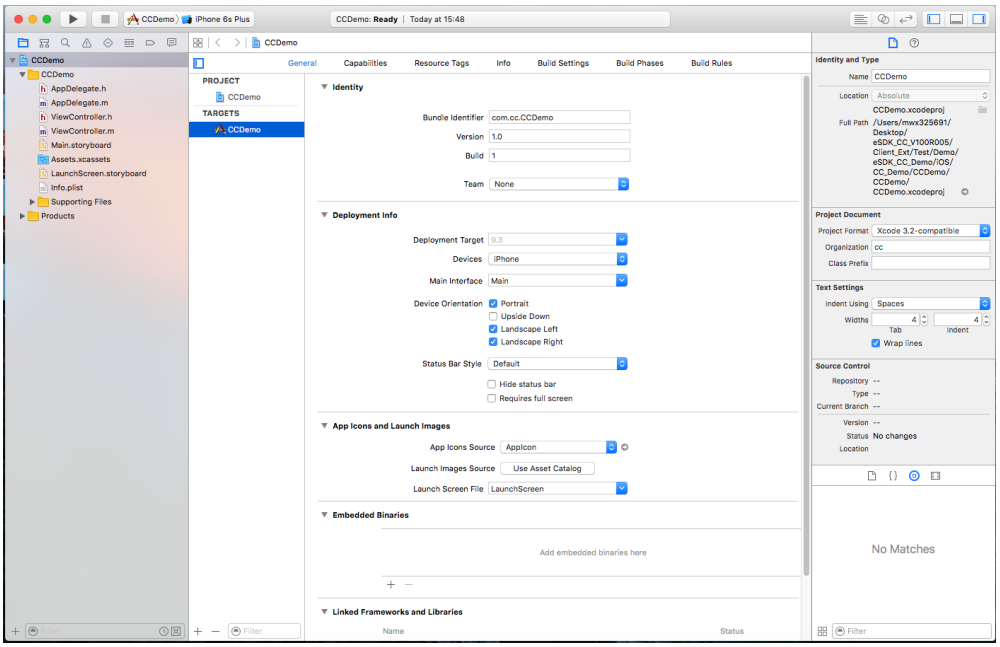
Step 1 Open the Xcode, choose **File** > **New** > **Project**, and click **Next**.



Step 2 Specify **Product Name, Organization Name**, and **Organization Identifier**, and click **Next**.



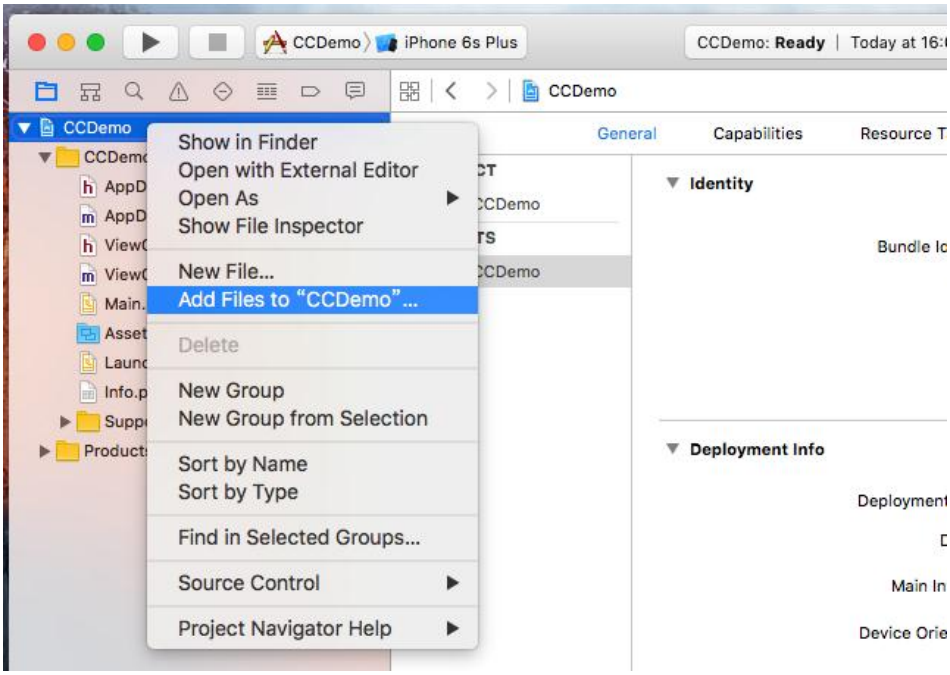
Step 3 Wait until project creation is completed. The following figure shows the page displayed after project creation.



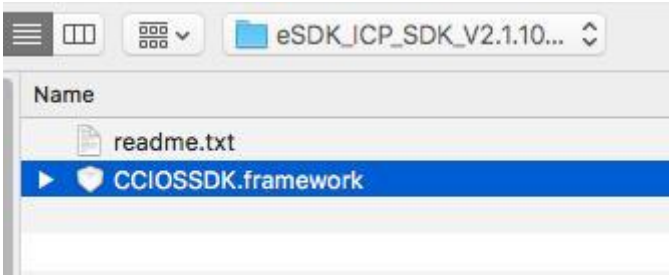
----End

4.4 Importing SDK into a Project

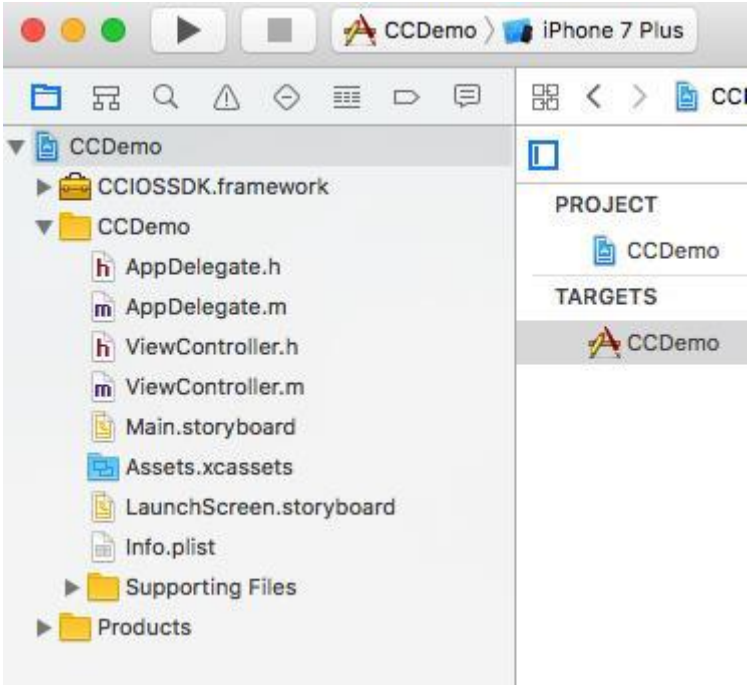
Step 1 Right-click a project and choose **Add Files to "CCDemo"**.



Step 2 Select a corresponding SDK folder.



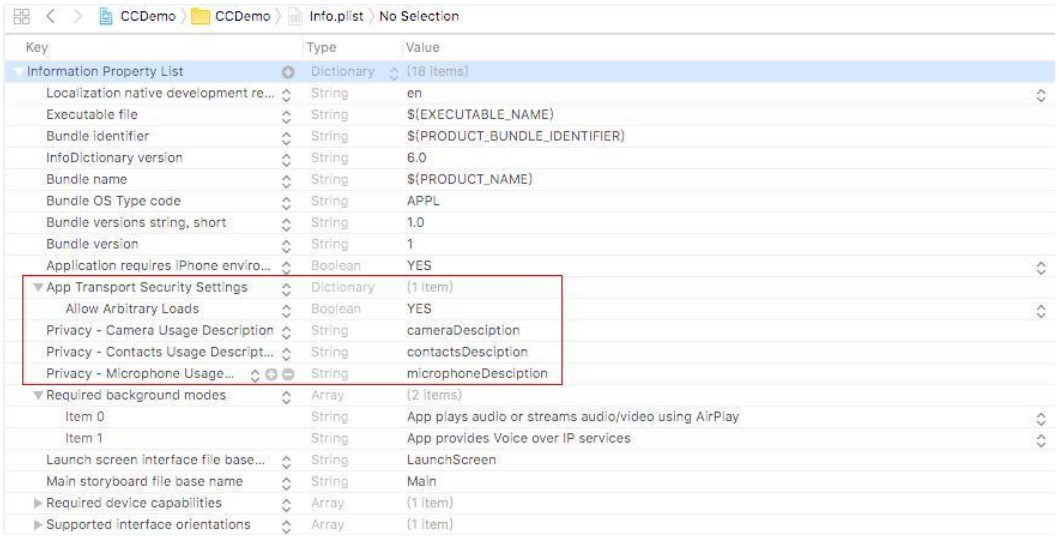
Step 3 Select the corresponding SDK, and click **Add**. The following figure shows the page displayed after successful adding.



----End

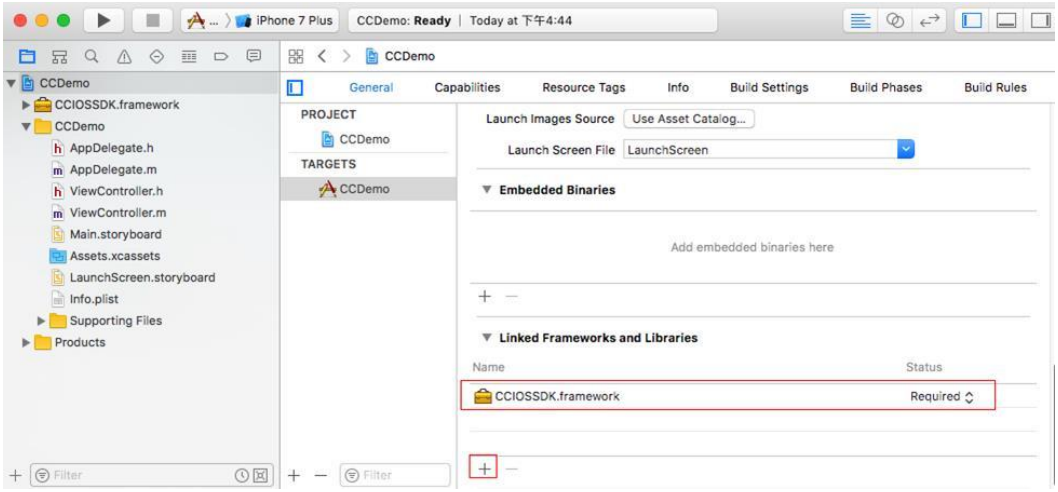
4.5 Configuring Other Projects

Step 1 To ensure successful login, select the **Info.plist** file under **Navigator** on the project main page, right-click any line, select **Add Row** (you can drop down to search one by one or input **App Transport Security Settings** directly for a search), and enter **Allow Arbitrary Loads** at the **Value** position of item0 and set the **Boolean** value to **YES**.

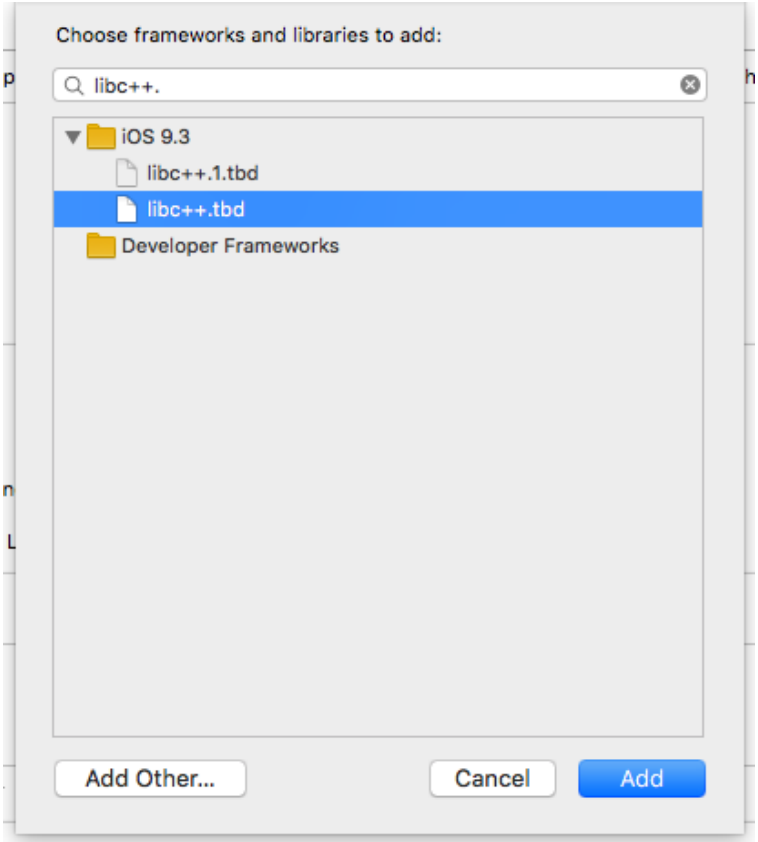


Step 2 Set Bitcode, click the project name under Navigator, and change the value of **Enable Bitcode** to **No**, as shown in the following figure.

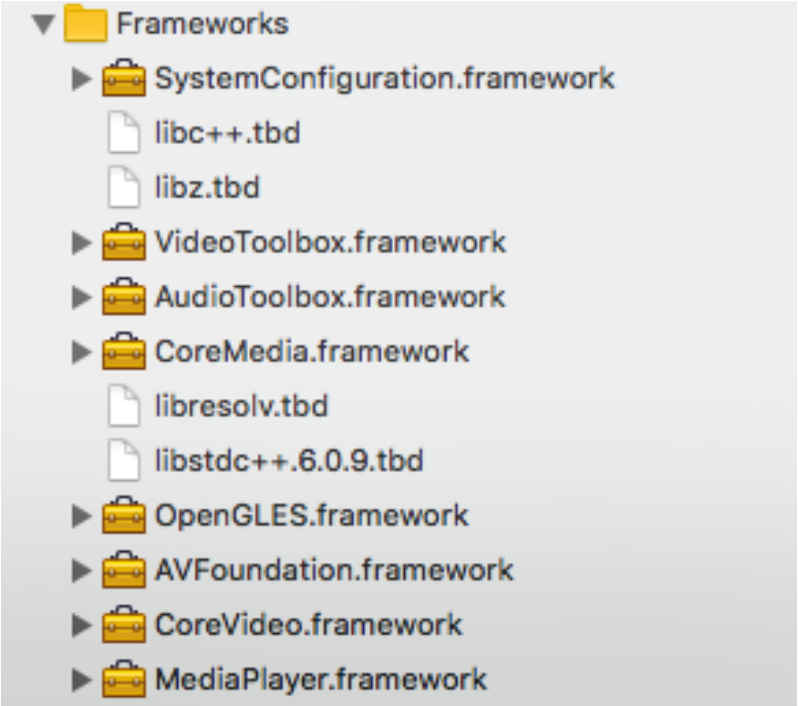
Step 3 Click the + button shown in the following figure successively. A dialog box of adding the dependence library file is displayed.



Step 4 Search for **libc++.tbd** through the search box, select it, and click **Add**.



Step 5 Repeat the preceding steps to add all the dependence frameworks shown in the following figure respectively.



----End

4.6 Implementing the Code

Overall Structure

```
CCDemo
----CCDemo
    ---------ViewController.h
    ---------ViewController.m
    ---------Main.storyboard
    ---------AppDelegate.h
    ---------AppDelegate.m
----CCIOSSDK.framework
```

Source code link: 8.1 Hello World Source Code File

Code Reference

The following provides the references for some key codes:

Registering login and logout notification

```
//objc code
- (void)addNotifions{
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(loginResult:)
name:AUTH_MSG_ON_LOGIN object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
```

```
selector:@selector(logoutResult:)
name:AUTH_MSG_ON_LOGOUT object:nil];
}
```

Setting the access gateway address and logging in

```
//objc code
NSInteger setRes = [[CCSDK shareInstance] setHostAddress:self.ipText.text
port:self.portText.text transSecurity:NO sipServerType:enType];
if (setRes != RET OK){
[self showMessage:[NSString stringWithFormat:@"%@:%ld",NSLocalizedString(@"Host",
""),(long)setRes]];
}
else{
NSInteger ret = [[CCSDK shareInstance] login:@"1" userName:self.userNameText.text];
```

4.7 Compiling and Commissioning

With Huawei CC Environment

If you have deployed Huawei ICP solution, fill in the user name, password, and IP address for logging in to the platform directly, and Commissioning and Running the Application the application.

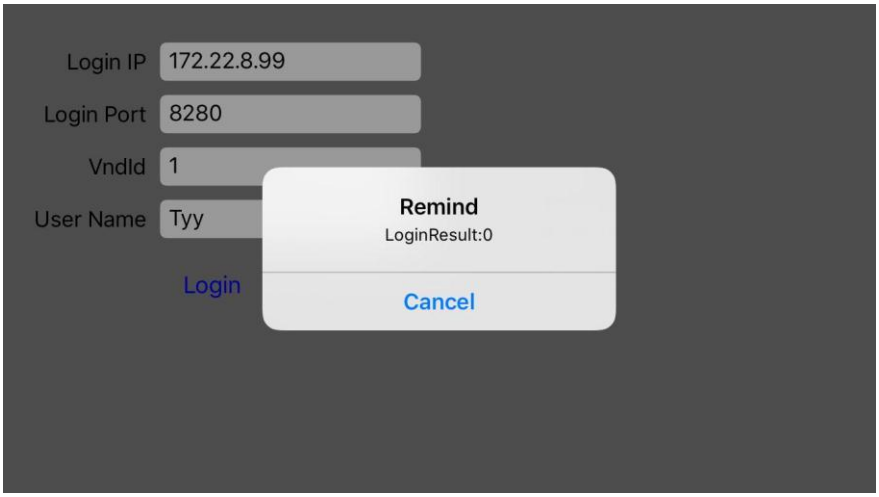
Without Huawei CC Environment

If you have not deployed Huawei ICP solution, log in to the 3.6 Free Application for the Remote Lab, apply for the ICP environment for free, and Commissioning and Running the Application the application.

Commissioning and Running the Application

Step 1 Choose **Product** > **Run**. The login screen is displayed.

Step 2 After entering relevant information, click **Login**. Wait until **Remind LoginResult:0** is displayed (this message indicates that the login is successful).



----End

5 Typical Development Scenarios

5.1 Scenario 1: Login and Logout

Function Description

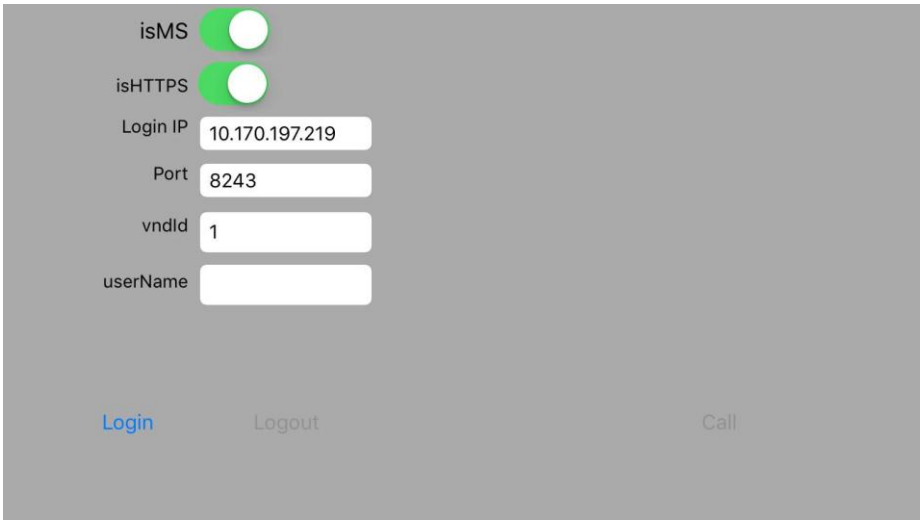
Used for login and logout.

For the case of logon instability, we need to set net.ipv4.tcp.timetamps to 0 in the server operating system

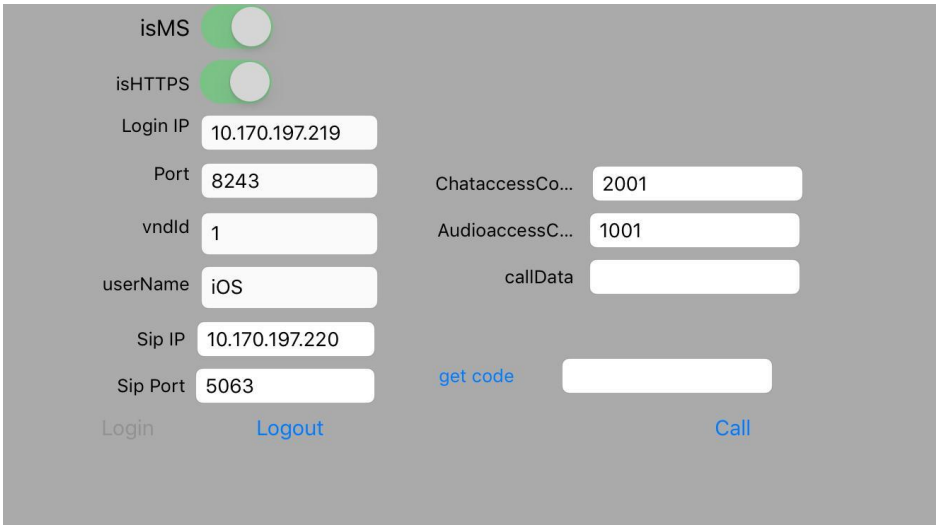
Sample Code

In demo mode, **LoginViewController.h** and **LoginViewController.m** are used for login and logout processing.

The following figure shows the login screen.



Login Success Page:



The following provides the key code. For the specific codes, see the **LoginViewController.m** file.

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

//Register notification
- (void)addNotifions{
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(loginResult:)
name:AUTH MSG ON LOGIN object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(logoutResult:)
name:AUTH MSG ON LOGOUT object:nil];
}

//Login result
- (void)loginResult:(NSNotification *)notificatio{
NSString *loginResult = (NSString *)notificatio.object;
if ([loginResult isEqualToString:@"0"]) {
```

```
dispatch_async(dispatch_get_main_queue(), ^{
[self loginSuccess];
});
}else{
dispatch_async(dispatch_get_main_queue(), ^{
[self showMessage:[NSString stringWithFormat:@"login failed:%@",loginResult]];
});
}
}

//Logout result
- (void)logoutResult:(NSNotification *)notification{
NSString *logoutResult  = (NSString *)notification.object;
if ([logoutResult isEqualToString:@"0"]) {
dispatch_async(dispatch_get_main_queue(), ^{
[self logoutSuccess];
});
}else{
dispatch_async(dispatch_get_main_queue(), ^{
[self showMessage:[NSString stringWithFormat:@"login failed:%@",logoutResult]];
});
}
}

//Login operation
- (IBAction)loginClick:(id)sender {
NSInteger setRes = [[CCSDK shareInstance] setHostAddress:self.ipText.text
port:self.portText.text transSecurity:NO];
if (setRes != RET OK){
[self showMessage:[NSString stringWithFormat:@"setting error:%ld",(long)setRes]];
}
else{
NSInteger ret = [[CCSDK shareInstance] login:@"1" userName:self.userNameText.text];
if (ret == RET OK) {
}else{
[self showMessage:[NSString stringWithFormat:@"login interface invocation
failure:%ld",(long)ret]];
};
}
```

The following table lists the main interfaces involved.

Interface	Description
• (NSInteger)setHostAddress:(NSString *)ip port:(NSString *)port transSecurity:(BOOL)transSec sipServerType:(int)serverType;	Used to set the login gateway address, select HTTP or HTTPS, and select the environment.
• (NSInteger)login:(NSString *)vndid userName:(NSString *)userName;	Used for login.
• (void)logout;	Used for logout.

5.2 Scenario 2: TP Call Function

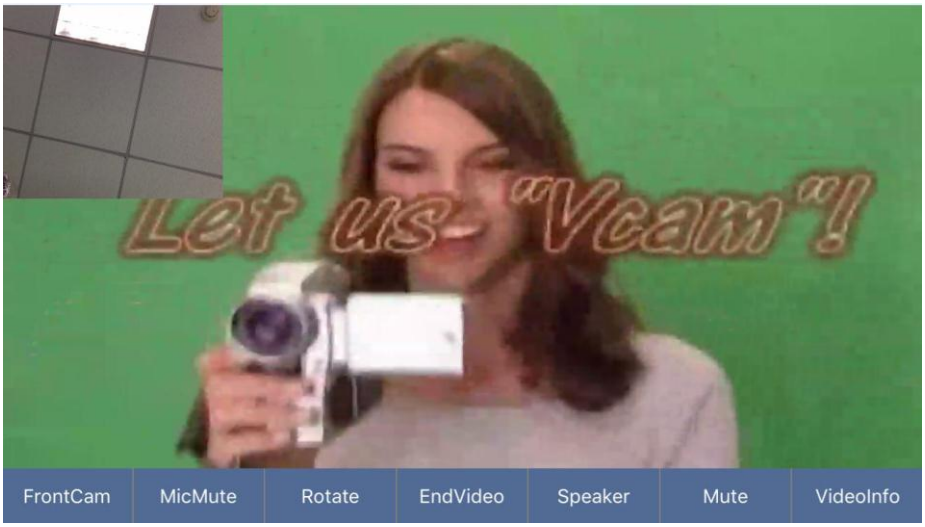
Function Description

This section describes the video call function.

Sample Code

The following figure shows call queuing.

The following figure shows successful call setup.



The following provides the key code. For the specific codes, see the **TPViewController.m** file.

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

//Register notification
- (void)addNotifications{
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(showVerifyCode:)
```

```
name:CALL_GET_VERIFY_CODE
object:nil];

[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(TPcallSuccess:)
name:CALL_MSG_ON_CONNECTED object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(TPcallEnd:)
name:CALL_MSG_ON_DISCONNECTED object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(TPcallFail:)
name:CALL_MSG_ON_FAIL object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(TPcallIsQueuing:)
name:CALL_MSG_ON_QUEUING object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(TPcallQueueTimeOut:)
name:CALL_MSG_ON_QUEUE_TIMEOUT
object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(TPreceiveQueInfo:)
name:CALL_MSG_ON_QUEUE_INFO object:nil];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(TPqueueIsCancel:)
name:CALL_MSG_ON_CANCEL_QUEUE object:nil];

}

//Process the verification code acquisition notification
- (void)showVerifyCode:(NSNotification *)notify
{
if ([notify.object intValue] == 0) {
NSString *encodedImageStr = [notify.userInfo objectForKey:@"verifyCode"];
NSData *decodedImageData =  [[NSData alloc]
initWithBase64EncodedString:encodedImageStr options:0];

UIImage *decodedImage= [UIImage imageWithData:decodedImageData];
dispatch_async(dispatch_get_main_queue(), ^{
//Throw back to the main thread for display
});
}
else{
dispatch_async(dispatch_get_main_queue(), ^{
//Throw back to the main thread for prompt. Acquisition failed.
});
}
}

//Obtain the verification code first
- (IBAction)refreshBtnClick:(id)sender {
[[CCSDK shareInstance] getVerifyCode];
}

//Call
- (void)callClick{
```

```
if (_callSuccess) {
[[CCSDK shareInstance] releaseCall];
}else{
NSInteger callRet = [[CCSDK shareInstance] makeCall:self.aCode callType:VIDEO_CALL
callData:self.callData verifyCode:@"5679"];
if (callRet != RET_OK) {
[self showMessage:[NSString stringWithFormat:@"call interface invocation
failure:%ld",(long)callRet]];
}else{
dispatch_async(dispatch_get_main_queue(), ^{
self.remindLabel.text = @"calling";
});
}
}
}

//Cancel queuing
- (void)cancelQueueClick
{
[[CCSDK shareInstance] cancelQueue];
dispatch_async(dispatch_get_main_queue(), ^{
[self.cancelQueueBtn removeFromSuperview];
});
}

//Obtain the queue information
- (void)getQueue{
[[CCSDK shareInstance] getCallQueueInfo];
}
```

The following table lists the main interfaces involved.

Interface	Description
• (BOOL)getVerifyCode	Used to obtain the verification code.
• - (NSInteger)makeCall:(NSString *)accessCode callType:(NSString *)callType callData:(NSString *)callData verifyCode：(NSString *)verifyCode;	Used to establish a voice/video call.
• (void)releaseCall;	Used to end a call.
• (void)getCallQueueInfo;	Used to obtain the queue information.
• (void)cancelQueue;	Used to cancel queuing.
• (void)setVideoContainer:(id)localView remoteView:(id)remoteView;	Used to set the local/remote video display container.

5.3 Scenario 3: MS Call Function

Function Description

The Chapter introduces text conversations, voice calls, conferencing, and desktop sharing in the MS environment. NAT traversal has been done in sdk, as long as the corresponding parameters can be configured.

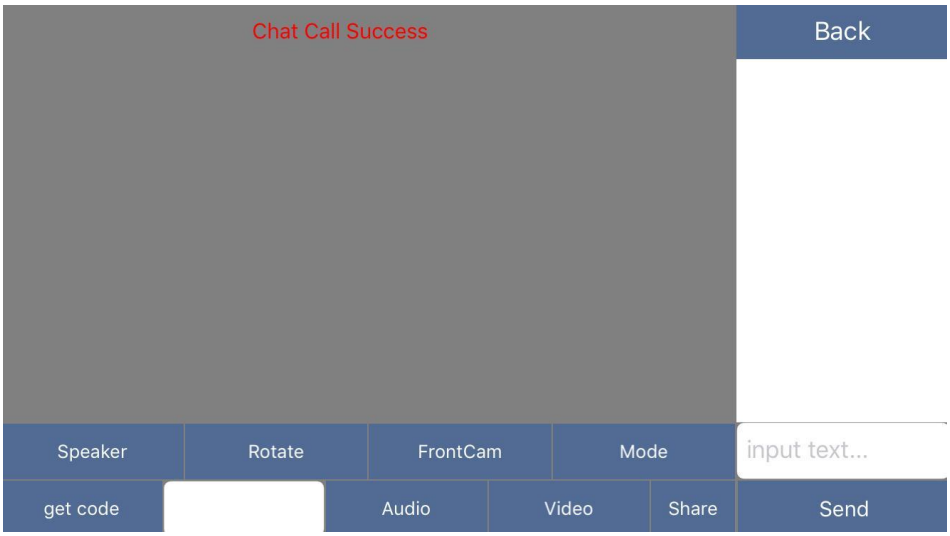
If the voice is connected immediately to disconnect the situation, please go to the router configuration, open the SIP ALG in the "WAN settings" .

If you do not need a verification code, you can open the home / prometheus / tomcat7 / webapps / icsgateway / WEB-INF / config / verifycode.properties file in the IcsGateway server, modify VERIFYCODE_ISUSERFORCALL = false, and then restart the IcsGateway server.

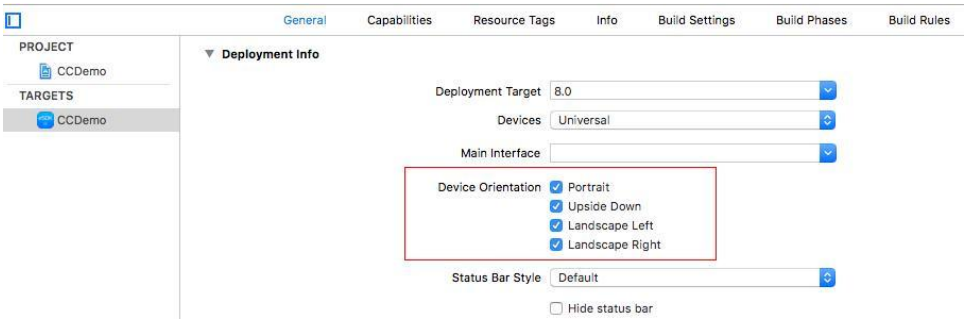
Sample Code

The call interface is shown below.

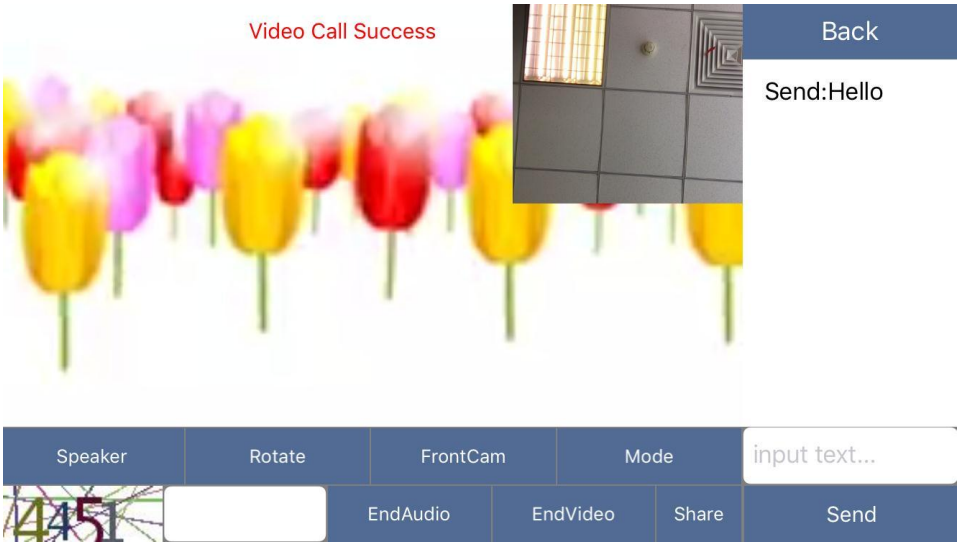
Text call:



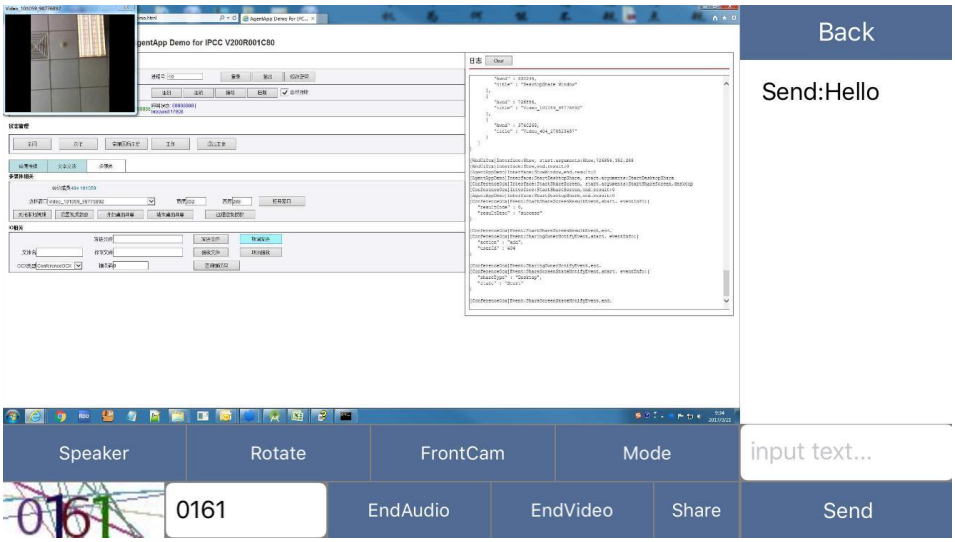
Video call settings (the local video interface settings for the vertical and horizontal screen are the sanme with the app settings , if the app all directions are supported, the video interface in the phone not locked horizontal or vertical will be adjusted according to gravity):



Successful video call:



Share the desktop as shown below:



The key code is shown below, and the specific code refers to the MSViewController.m file in eSDK_ICP_Demo_V2.1.10_iOS.

```
//引入头文件
 #import <CCIOSSDK/CCIOSSDK.h>

//注册通知- (void)addNotifications{

    [[NSNotificationCenter defaultCenter] addObserver:self
          selector:@selector(callSuccess:)
          name:CALL MSG ON CONNECTED object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
           selector:@selector(callEnd:)
           name:CALL MSG ON DISCONNECTED object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
           selector:@selector(userLeave:)
           name:CALL_MSG_ON_USER_LEAVE object:nil];
```

```
    [[NSNotificationCenter defaultCenter] addObserver:self
          selector:@selector(callFail:)
          name:CALL_MSG_ON_FAIL object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
          selector:@selector(sendMsgSuccess)
          name:CHAT_MSG_ON_SUCCESS object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
          selector:@selector(sendMsgFail:)
          name:CHAT_MSG_ON_FAIL object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
          selector:@selector(receiveMessage:)
          name:CHAT_MSG_ON_RECEIVE object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
          selector:@selector(callIsQueuing:)
          name:CALL_MSG_ON_QUEUING object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
            selector:@selector(queueIsCancel:)
            name:CALL MSG ON CANCEL QUEUE object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
            selector:@selector(queueTimeOut:)
            name:CALL MSG ON QUEUE TIMEOUT object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
            selector:@selector(receiveQueueInfo:)
            name:CALL MSG ON QUEUE INFO object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
            selector:@selector(receiveDeskShare:)
            name:CALL MSG ON SCREEN DATA RECEIVE object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
            selector:@selector(deskShareStop:)
            name:CALL MSG ON SCREEN SHARE STOP object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self
            selector:@selector(showVerifyCode:)
            name:CALL GET VERIFY CODE object:nil];
}
//获取验证码

- (void)refreshBtnClick{
    [self.refreshBtn setBackgroundColor:[UIColor clearColor]];
    [[CCSDK shareInstance] getVerifyCode];
}
- (void)showVerifyCode:(NSNotification *)notify
{
    if ([notify.object intValue] == 0) {
        NSString *encodedImageStr = [notify.userInfo objectForKey:@"verifyCode"];
```

```
        NSData *decodedImageData   =  [[NSData alloc]
initWithBase64EncodedString:encodedImageStr options:0];

        UIImage *decodedImage= [UIImage imageWithData:decodedImageData];

        NSLog(@"===Decoded image size: %@", NSStringFromCGSize(decodedImage.size));
        dispatch_async(dispatch_get_main_queue(), ^{
            [self.verifyCodeImg setImage:decodedImage];
            [self.refreshBtn setTitle:@"" forState:UIControlStateNormal];
        });
    }
    else{
        dispatch_async(dispatch_get_main_queue(), ^{
            [CCDemoUtil showAlertWithTitle:NSLocalizedString(@"Remind", "")
                                content:NSLocalizedString(@"GetCodeError", "")];
        });


    }

//发起文字交谈
- (void)doWebChatCall
{
    [[CCSDK shareInstance] webChatCall:[LoginInfo sharedInstance].MSChatACode
callData:[LoginInfo sharedInstance].MSCallData verifyCode:[LoginInfo
sharedInstance].VCode];
     isWebCall = YES;
    /* callTimer = [NSTimer scheduledTimerWithTimeInterval:1
                                          target:self
                                        selector:@selector(countTimer)
                                        userInfo:nil repeats:YES];*/
    [self startCallTimer];
    self.remindLabel.text = [NSString
stringWithFormat:@"%@ %@",NSLocalizedString(@"Chat",
""),NSLocalizedString(@"Calling", "")];
     isCalling = YES;
}
//发送文字消息
- (void)sendMessageClick
{

    if ( isCalling)
    {
        [CCDemoUtil showAlertWithTitle:NSLocalizedString(@"Remind", "")
content:NSLocalizedString(@"Current", "")];
        return;
    }
    if (self.chatText.text.length == 0) {
        [CCDemoUtil showAlertWithTitle:NSLocalizedString(@"Remind", "")
content:NSLocalizedString(@"NoMessage", "")];
        return;
    }

    NSInteger ret = [[CCSDK shareInstance] sendMsg:self.chatText.text];
    if (ret != RET_OK)
```

```
    {
        [CCDemoUtil showAlertWithTitle:NSLocalizedString(@"Remind", "")
                                content:[NSString
stringWithFormat:@"%@:%ld",NSLocalizedString(@"SendInterface", ""),(long)ret]];
        return;
    }
    NSString *message = [NSString stringWithFormat:@"%@:%@",NSLocalizedString(@"Send",
""),self.chatText.text];
    [self.dataArray addObject:message];
    self.chatText.text = @"";
    [self.chatTableView reloadData];
    [self.chatTableView scrollToRowAtIndexPath:[NSIndexPath
indexPathForRow:self.dataArray.count - 1 inSection:0]
atScrollPosition:UITableViewScrollPositionBottom animated:YES];
}

//视频呼叫
- (void)videoCallClick
{
    if (_isCalling)
    {
        [CCDemoUtil showAlertWithTitle:NSLocalizedString(@"Remind", "")
content:NSLocalizedString(@"Current", "")];
        return;
    }


    if ( videoSuccess)
    {
        [[CCSDK shareInstance] releaseCall];
         videoSuccess = NO;
        if ( screenShare) {
            screenShare = NO;
        }
        if ( videoViewOpen)
        {
            videoViewOpen = NO;
            dispatch_async(dispatch_get_main_queue(), ^{
                [self closeView];
            });
        }
        if ( shareViewOpen)
        {
            shareViewOpen = NO;
            dispatch_async(dispatch_get_main_queue(), ^{
                [self.screenshareView removeFromSuperview];
            });
        }
        dispatch_async(dispatch_get_main_queue(), ^{
            [self.remindLabel removeFromSuperview];
            self.remindLabel.text =[NSString
stringWithFormat:@"%@ %@",NSLocalizedString(@"Video",
""),NSLocalizedString(@"CallEnd", "")];
            [self.view addSubview:self.remindLabel];
            [self.videoBtn setTitle:NSLocalizedString(@"Video", "")
```

```
forState:UIControlStateNormal];
       });
    }
    else
    {
        _isCalling = YES;
       if(_audioSuccess)
       {
           [self startCallTimer];
           self.remindLabel.text = [NSString
stringWithFormat:@"%@ %@",NSLocalizedString(@"Video",
""),NSLocalizedString(@"Calling", "")];
           [[CCSDK shareInstance] updateToVideo];
       }
       else
       {
           self.remindLabel.text = [NSString
stringWithFormat:@"%@ %@",NSLocalizedString(@"Video",
""),NSLocalizedString(@"Calling", "")];
           [[CCSDK shareInstance] makeCall:[LoginInfo sharedInstance].MSAudioACode
callType:VIDEO_CALL callData:[LoginInfo sharedInstance].MSCallData
verifyCode:self.verifyText.text];
       }
    }
}

//语音呼叫
-(void)audioCall
{
     isCalling = YES;

    self.remindLabel.text = [NSString
stringWithFormat:@"%@ %@",NSLocalizedString(@"Audio",
""),NSLocalizedString(@"Calling", "")];
    if (! audioSuccess)
    {
       [[CCSDK shareInstance] makeCall:[LoginInfo sharedInstance].MSAudioACode
callType:AUDIO_CALL callData:[LoginInfo sharedInstance].MSCallData
verifyCode:self.verifyText.text];
    }
    else
    {
       [[CCSDK shareInstance] releaseCall];
       if ( videoSuccess)
       {
            videoSuccess = NO;

           if ( screenShare)
           {
               screenShare = NO;
           }
           if ( videoViewOpen)
           {
               videoViewOpen = NO;
              dispatch_async(dispatch_get_main_queue(), ^{
```

```
                [self closeView];
            });
        }

        if (_shareViewOpen)
        {
            _shareViewOpen = NO;
            dispatch_async(dispatch_get_main_queue(), ^{
                [self.screenshareView removeFromSuperview];
            });
        }
        dispatch_async(dispatch_get_main_queue(), ^{
            [self.remindLabel removeFromSuperview];
            self.remindLabel.text = NSLocalizedString(@"CallEnd", "");
            [self.view addSubview:self.remindLabel];
            [self.videoBtn setTitle:NSLocalizedString(@"Video", "")
forState:UIControlStateNormal];
        });
    }
  }
}
//设置桌面共享容器
 [[CCSDK shareInstance] setDesktopShareContainer:self.screenshareView];
```

The following table lists the main interfaces involved.

Interface	Description
- (NSInteger)setSIPServerAddress:(NSString *)ip port:(NSString *)port ;	Set the SIP server address (call address)
- (NSInteger)webChatCall:(NSString *)accessCode callData:(NSString *)callData verifyCode:(NSString *)verifyCode;	Initiate a text conversation
- (NSInteger)sendMsg:(NSString *)message;	Send a text message
- (NSInteger)makeCall:(NSString *)accessCode callType:(NSString *)callType callData:(NSString *)callData verifyCode:(NSString *)verifyCode;	Voice / video call
- (NSInteger)updateToVideo;	Voice upgrade to video
- (void)releaseCall;	End call
- (void)getCallQueueInfo;	Get queuing information
- (void)cancelQueue;	Cancel queuing
- (void)setVideoContainer:(id)localView remoteView:(id)remoteView;	Set the local and remote video display containers
- (void)setDesktopShareContainer:(UIImage View *)shareView;	Set up a desktop sharing container

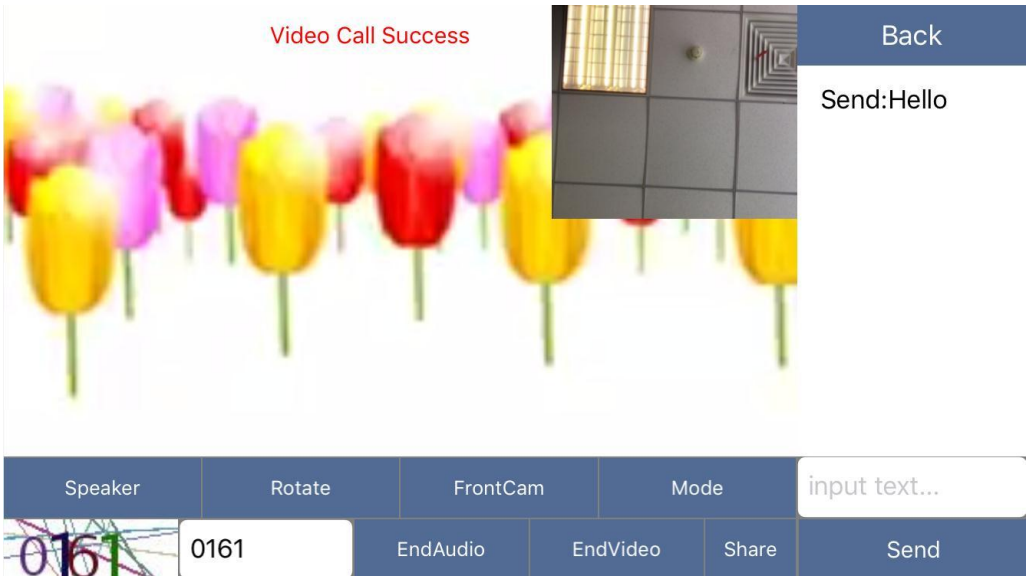
5.4 Scenario 4: Device Control

Function Description

This section describes how to switch between the front-facing camera and rear-facing camera, switch between the speaker mode and earpiece mode, and mute the microphone during a call.

Sample Code

The following figure shows the screen for using the camera, microphone muting, and speaker.



The following provides the key code. For the specific codes, see the **TPViewController.m** file.

```
//Mute the microphone
- (void)setMicMuteClick{
if (! callSuccess)
{
[self showMsg:NSLocalizedString(@"NoVideo", "")];
return;
}
if ( micisMute) {
[[CCSDK shareInstance] setMicMute:NO];
[self.micMuteBtn setTitle:NSLocalizedString(@"MicMute", "")
forState:UIControlStateNormal];
 micisMute = NO;
} else {
[[CCSDK shareInstance] setMicMute:YES];
[self.micMuteBtn setTitle:NSLocalizedString(@"OpenMic", "")
forState:UIControlStateNormal];
 micisMute = YES;
}
}
```

```
//Switch the camera
- (void)switchCameraClick{
if (!_callSuccess)
{
[self showMsg:NSLocalizedString(@"NoVideo", "")];
return;
}
[[self class] cancelPreviousPerformRequestsWithTarget:self
selector:@selector(cameraSwitch) object:nil];
[self performSelector:@selector(cameraSwitch) withObject:nil afterDelay:0.7];
}
- (void)cameraSwitch{
_rotate = 0;
if (_isBackCamera) {
[[CCSDK shareInstance] switchCamera:1];
_isBackCamera = NO;
}else{
[[CCSDK shareInstance] switchCamera:0];
_isBackCamera = YES;
}
}

//Swtich between the speaker mode and earpiece mode
- (void)changeAudioRouteClick{
if (! callSuccess)
{
[self showMsg:NSLocalizedString(@"NoVideo", "")];
return;
}
if (! speakisMute) {
[[CCSDK shareInstance] changeAudioRoute:0];
[self.speakMuteBtn setTitle:NSLocalizedString(@"Speaker", "")
forState:UIControlStateNormal];
 speakisMute = YES;
} else {
[[CCSDK shareInstance] changeAudioRoute:1];
[self.speakMuteBtn setTitle:NSLocalizedString(@"Receiver", "")
forState:UIControlStateNormal];
 speakisMute = NO;
}
}
```

The following table lists the main interfaces involved.

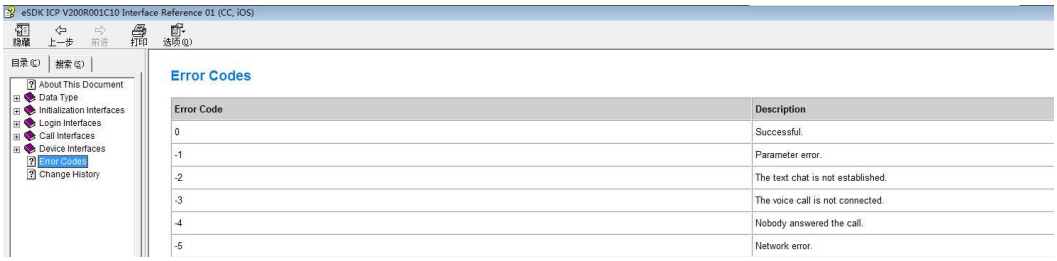
Interface	Description
- (BOOL)changeAudioRoute:(int)route	Speaker / handset mode switch
-(BOOL)setVideoRotate:(VIDEO_ROTATE)rotate	Set the video rotation angle
- (BOOL)switchCamera:(int)index	Front / rear camera switch
- (BOOL)setVideoMode:(int)videoMode	Set the video display mode

6 Fault Locating Guide

Querying Error Information

The interface reference describes all error codes.

You can query error information based on the error code.



Obtaining Logs

Returned upon interface invocation

After an interface is invoked, a value is returned. If the return value is **0**, the interface is successfully invoked. Otherwise, the interface fails to be invoked, and the return value is the error code.

Obtained from the log files

You can obtain the log file based on the log path imported through the interface for setting the log path.

Analyzing Logs

The following takes the method of invoking the interface for setting the gateway as an example. Search for the keyword **setHostAddress** globally to obtain a corresponding record. If this record ends with |0|, this interface is invoked successfully.

```
2016-08-17 11:15:27 869| INFO|eSDK-CC-API-iOS|1|Native|setHostAddress||||2016-08-17
11:15:27 567|2016-08-17 11:15:27 868|IPStr=172.22.9.40, portStr=8280,
transSecurity=false, sipServerType=1|0|
```

If this record ends with **|-1|**, this interface fails to be invoked.

```
2016-08-17 11:34:05 573|ERROR|eSDK-CC-API-iOS|1|Native|setHostAddress||||2016-08-17
11:34:05 570|2016-08-17 11:34:05 572|IPStr=172.22.9, portStr=8280,
transSecurity=false, sipServerType=1|-1|
```

7 Change History

Date	Issue	Description
2017-3-20	V2.1.10	Added: Document V200R001C10 is released. Full fit TP & MS function.
2016-12-31	V2.1.00	This issue is the first official release.

8 Appendix

8.1 Hello World Source Code File

8.1.1 ViewController.m

```
//objc code
#import "ViewController.h"
#import <CCIOSSDK/CCIOSSDK.h>

@interface ViewController ()

@property (strong, nonatomic)  UITextField *ipText;
@property (strong, nonatomic)  UITextField *portText;
@property (strong, nonatomic)  UITextField *userNameText;
@property (strong, nonatomic)  UITextField *vndIdText;

@property (strong, nonatomic) UIButton *loginBtn;

@end

@implementation ViewController

- (void)viewDidLoad {
[super viewDidLoad];
// Do any additional setup after loading the view, typically from a nib.
[self addNotifications];
[self initView];

}

- (void)didReceiveMemoryWarning {
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}

- (void)addNotifications
```

```
{

[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(onKeyBoardWillShow:)
name:UIKeyboardWillShowNotification
object:nil];

[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(onKeyBoardWillHide:)
name:UIKeyboardWillHideNotification
object:nil];

[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(loginResult:)
name:AUTH_MSG_ON_LOGIN
object:nil];

}

-(void)onKeyBoardWillShow:(NSNotification *)notify
{
NSDictionary *userInfo = [notify userInfo];
NSNumber *duration = userInfo[UIKeyboardAnimationDurationUserInfoKey];
[UIView animateWithDuration:[duration doubleValue] animations:^{
self.view.frame = CGRectMake(0, -50, [UIScreen mainScreen].bounds.size.width,
[UIScreen mainScreen].bounds.size.height);
}];
}
-(void)onKeyBoardWillHide:(NSNotification *)notify
{
NSDictionary *userInfo = [notify userInfo];
NSNumber *duration = userInfo[UIKeyboardAnimationDurationUserInfoKey];
[UIView animateWithDuration:[duration doubleValue] animations:^{
self.view.frame = CGRectMake(0, 0, [UIScreen mainScreen].bounds.size.width, [UIScreen
mainScreen].bounds.size.height);
}];
}

- (void)touchesBegan:(NSSet<UITouch *> *)touches withEvent:(UIEvent *)event{
[self.view endEditing:YES];
}

-(void)initView{
self.view.backgroundColor = [UIColor grayColor];
UILabel *loginIp = [[UILabel alloc] initWithFrame:CGRectMake(10, 30, 100, 30)];
loginIp.text=@"Login IP";
loginIp.textAlignment = NSTextAlignmentRight;
[self.view addSubview:loginIp];

self.ipText = [[UITextField alloc] initWithFrame:CGRectMake(120,30,  200, 30)];
self.ipText.placeholder = @"Login IP";
self.ipText.text = @"172.22.8.99";
self.ipText.borderStyle = UITextBorderStyleRoundedRect;
[self.view addSubview:self.ipText];
```

```
UILabel *loginPort = [[UILabel alloc] initWithFrame:CGRectMake(10, 70, 100, 30)];
loginPort.text=@"Login Port";
loginPort.textAlignment = NSTextAlignmentRight;
[self.view addSubview:loginPort];

self.portText = [[UITextField alloc] initWithFrame:CGRectMake(120,70, 200, 30)];
self.portText.placeholder = @"Login Port";
self.portText.text = @"8280";
self.portText.borderStyle = UITextBorderStyleRoundedRect;
[self.view addSubview:self.portText];

UILabel *loginVndId = [[UILabel alloc] initWithFrame:CGRectMake(10, 110, 100, 30)];
loginVndId.text=@"VndId";
loginVndId.textAlignment = NSTextAlignmentRight;
[self.view addSubview:loginVndId];

self.vndIdText = [[UITextField alloc] initWithFrame:CGRectMake(120,110,  200, 30)];
self.vndIdText.placeholder = @"VndId";
self.vndIdText.text = @"1";
self.vndIdText.borderStyle = UITextBorderStyleRoundedRect;
[self.view addSubview:self.vndIdText];

UILabel *userName = [[UILabel alloc] initWithFrame:CGRectMake(10, 150, 100, 30)];
userName.text=@"User Name";
userName.textAlignment = NSTextAlignmentRight;
[self.view addSubview:userName];

self.userNameText = [[UITextField alloc] initWithFrame:CGRectMake(120,150, 200, 30)];
self.userNameText.placeholder = @"User Name";
self.userNameText.borderStyle = UITextBorderStyleRoundedRect;
[self.view addSubview:self.userNameText];

self.loginBtn = [[UIButton alloc] initWithFrame:CGRectMake(120,200, 80, 30)];
[self.loginBtn setTitle:@"Login" forState:UIControlStateNormal];
[self.loginBtn setTitleColor:[UIColor blueColor] forState:UIControlStateNormal];
[self.loginBtn addTarget:self action:@selector(loginClick)
forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:self.loginBtn];

}

- (void)loginClick {
if ([self.userNameText.text length] == 0)
{
[self showAlertWithTitle:NSLocalizedString(@"Remind", "") content:@"userName is nil"];
return;
}

int serverType = SERVER_TYPE_TP;

[[CCSDK shareInstance] setHostAddress:self.ipText.text
port:self.portText.text
transSecurity:NO
sipServerType:serverType];
```

```
NSInteger result =  [[CCSDK shareInstance] login:self.vndIdText.text
userName:self.userNameText.text];
if (result != RET_OK)
{
[self showAlertWithTitle:NSLocalizedString(@"Remind", "") content:@"userName is
inValid"];
}
}

- (void)loginResult:(NSNotification *)notify
{
NSString *loginResult = (NSString *)notify.object;
dispatch_async(dispatch_get_main_queue(), ^{
[self showAlertWithTitle:NSLocalizedString(@"Remind", "")
content:[NSString stringWithFormat:@"%@:%@",NSLocalizedString(@"LoginResult",
""),loginResult]];
});
}

- (void)showAlertWithTitle:(NSString*)title content:(NSString*)content
{
UIAlertView *alertView = [[UIAlertView alloc]initWithTitle:title message:content
delegate:nil cancelButtonTitle:@"Cancel" otherButtonTitles:nil];
[alertView show];
}

@end
```