eSDK ICP V200R001C10 Development Guide 01 (CC, Android)

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. For details, see the 3.2 SDK Download Path.

- **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 Android interface-related services. For details, see the 3.3 Sample Code.

2 Development Guide Overview

This document provides guidance for developers to install and configure the eSDK CC Android environment, invoke the eSDK CC Android 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 Android, 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 Android services, read 5 Typical Development Scenarios.
- If you encounter any problems when you use the SDK, refer to 6 Fault Locating Guide or 5 Typical Development Scenarios.

3 Related Resources

3.1 Huawei Developer Zone

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

3.2 SDK Download Path

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

The latest V2.1.10 version is recommended.

3.3 Sample Code

You are advised to use Android Studio 1.5 to compile or execute the sample code.

The following provides the sample code list.

Demo	Description
eSDK_CC_Demo_V2.1.10_Android.zip	Typical scenario demo developed based on the eSDK CC Android Sample, including the call

Demo	Description
	and device control functions.

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 Free Application for the Remote Lab

Huawei eSDK Remote Lab Introduction

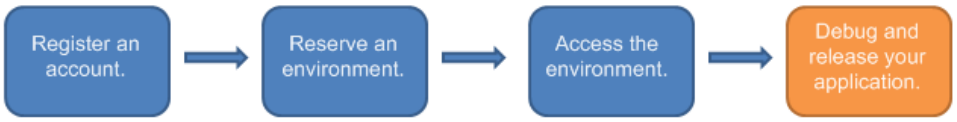
The Huawei remote lab provides 24/7 free cloud lab environment and real Huawei devices for developers to develop and commission applications online remotely. Using the remote lab self-management platform, developers can implement secondary development related to Huawei products without the need to purchase 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, enterprise cloud communication, and enterprise mobile security.

For details, visit the remote lab homepage.

Advantages of the Remote Lab

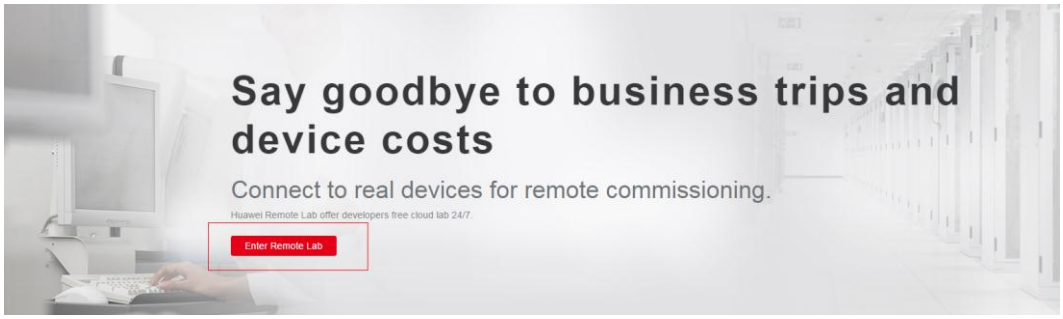
- Low entry barrier: Users who have been registered on the Huawei official website can apply to use the remote lab. Note that the environments that can be reserved, reservation duration, and the number of times the environments can be reserved are restricted.
- Hierarchical support: Environments are 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

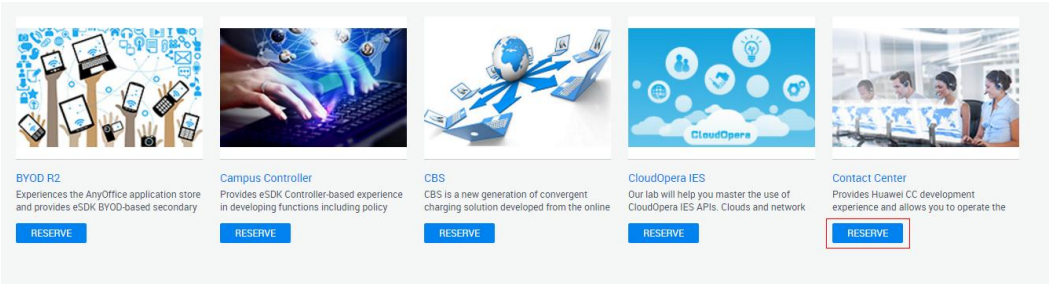


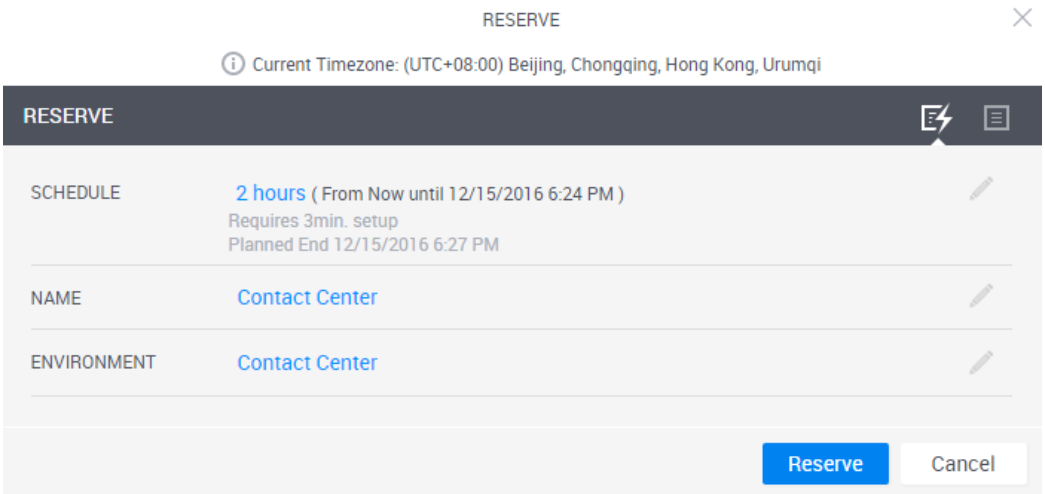
Step 1 Log in.

1. If you have a Huawei official website account, log in to the remote lab directly using the account.
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.

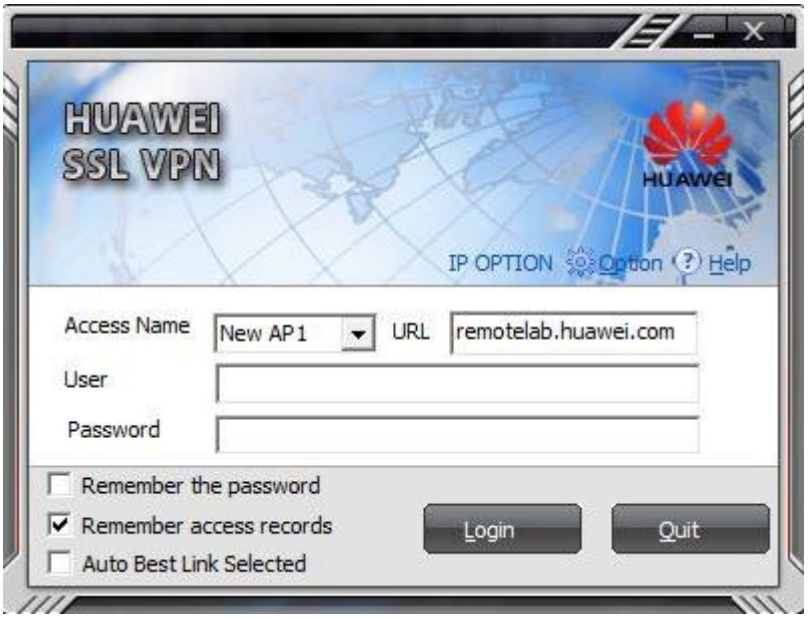




4. After you successfully reserve the environment, the system automatically sends the Secure Sockets Layer virtual private network (SVN) gateway address, user name, and password to your registration's email address. Log in to your registration's email account, open the email of the environment information, 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

3.6 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

3.7 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

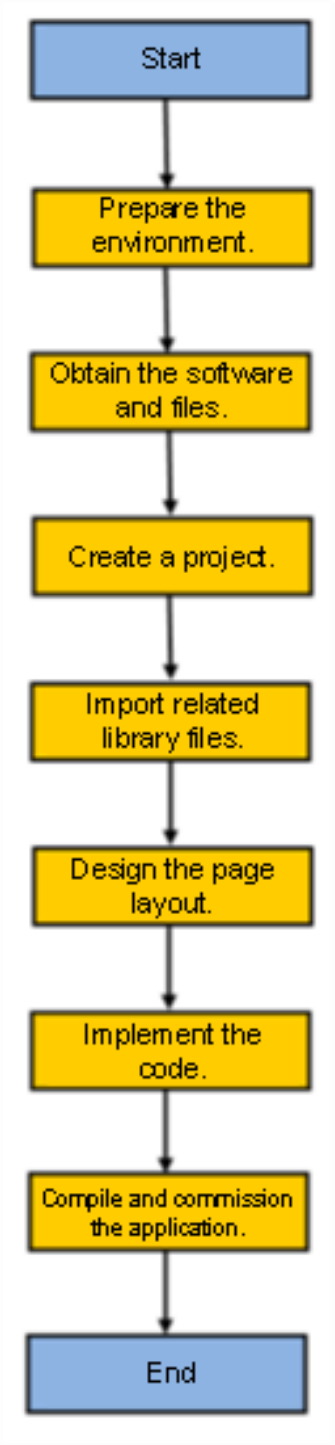
4.1 Overview

Hello World Development Process

The following example describes how to perform eSDK CC secondary development in Java.

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: Windows 7 Professional
- Android Studio 1.5 or later

- Java Development Kit 1.7 or later

SDK Software Package

- SDK software package name: **eSDK_CC_SDK_V2.1.10_Android.zip**
- SDK software package download path: See 3.2 SDK Download Path.

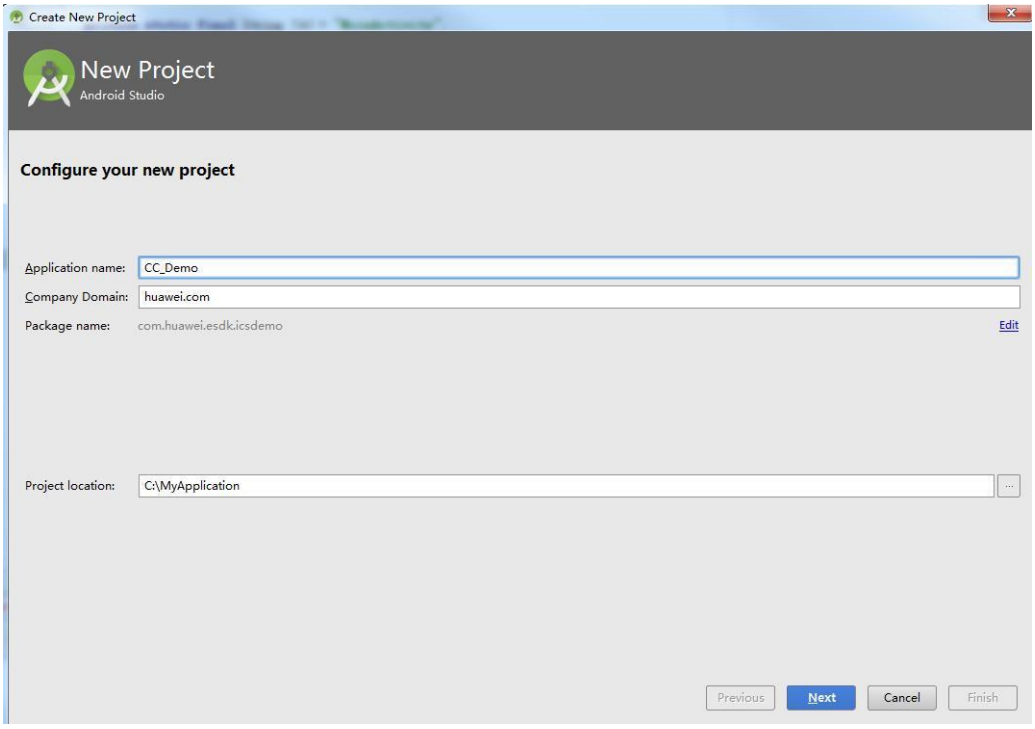
If the certificate verification function is required, the certificate needs to be placed in the **assets** folder of the project, for example, **assets/certs/server.cer**.

Using the Samsung note4 test call feature, you need to add Bluetooth permissions to your AndroidManifest file. The default network access is https mode. During the video call process, the current Activity set to horizontal screen, that is, android: screenOrientation = "landscape", the video is displayed horizontal, Actvity set to vertical screen, that is, android: screenOrientation = "portrait", the video is vertical of.

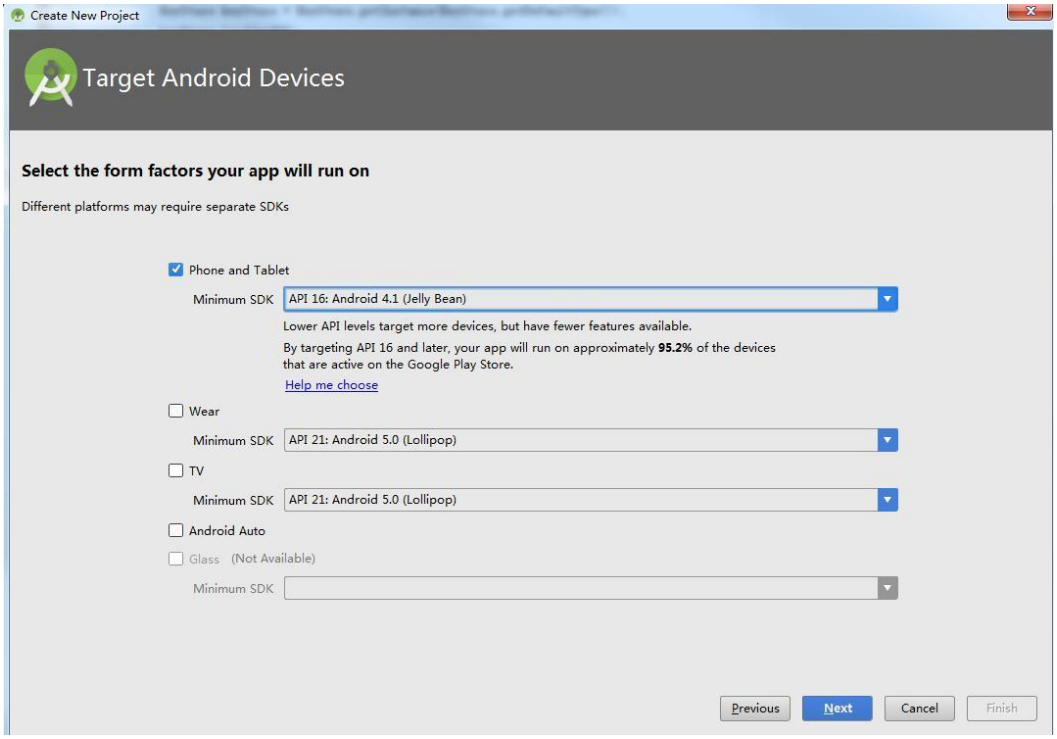
Using the local broadcast LocalBroadcastManager, you need to write 'com.android.support:support-v4:23.1.1' in the build.gradle dependency.

4.3 Creating a Project

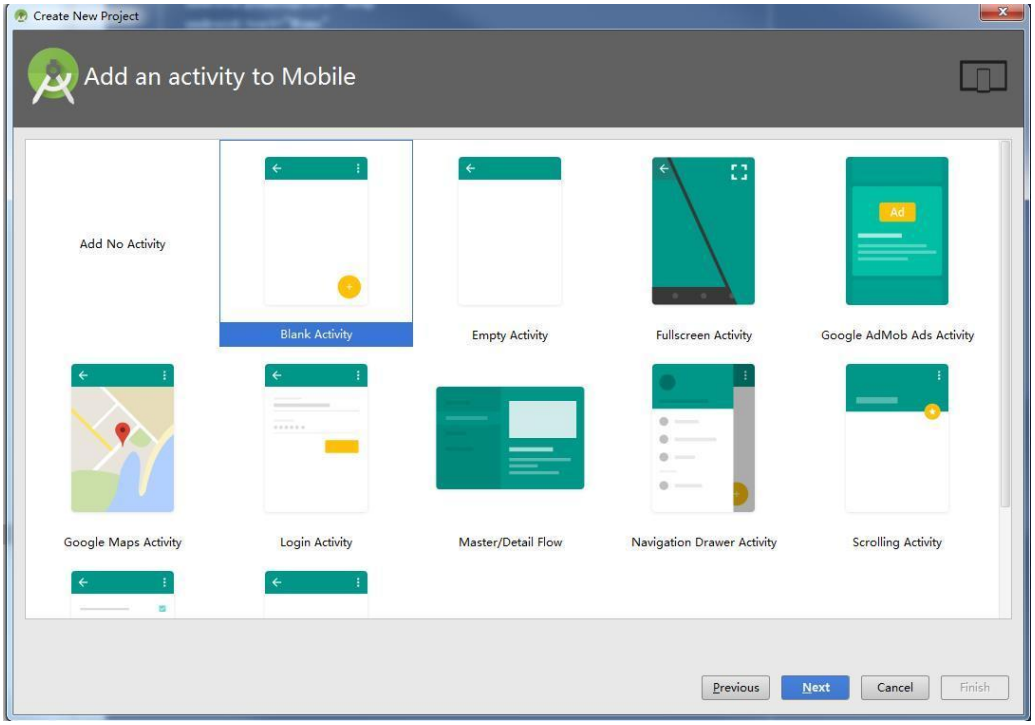
Step 1 Open the Android Studio and choose **File** > **New** > **New Project**. The **Create New Project** window is displayed.



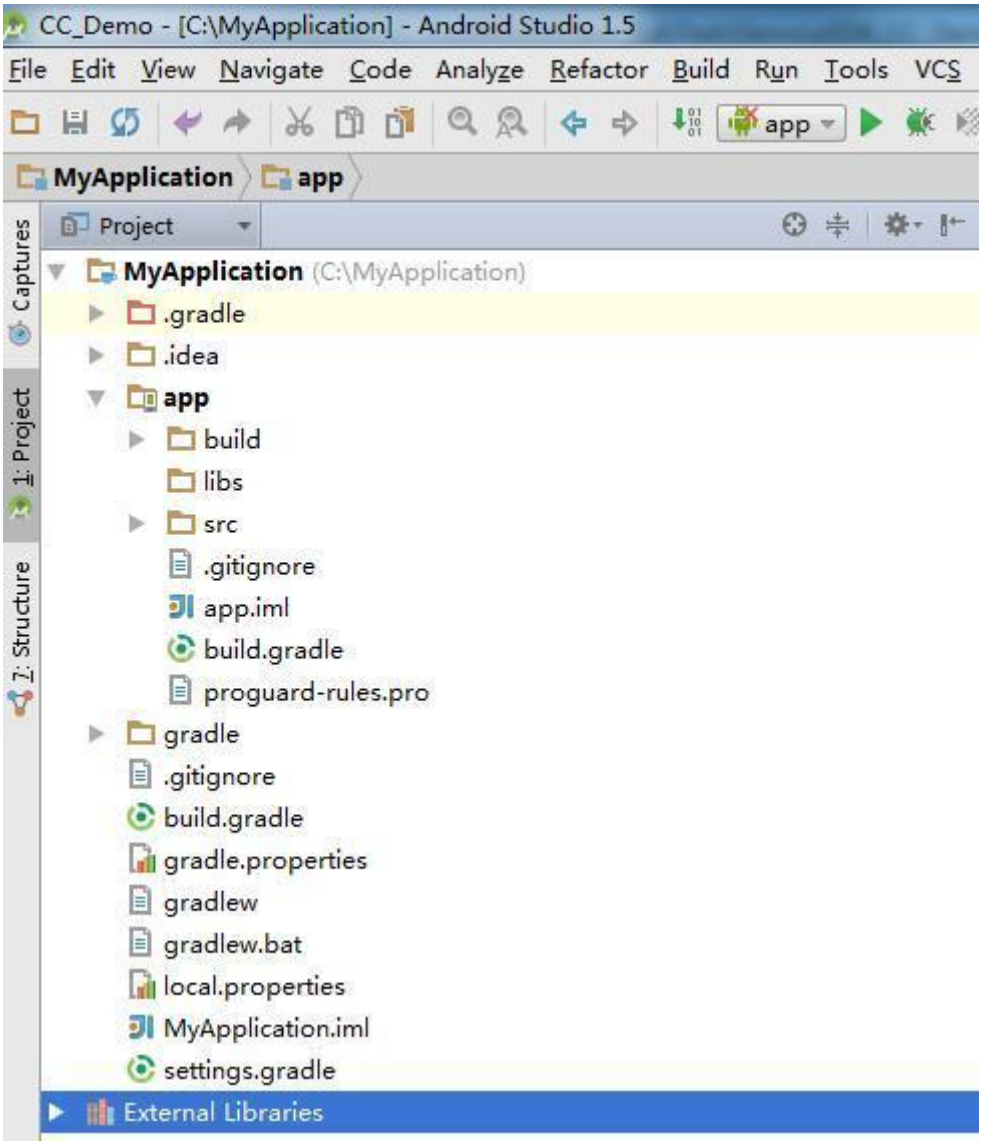
Step 2 Specify **Application name**, **Company Domain**, and **Project location**, and click **Next**. The SDK version selection page is displayed.



Step 3 Select the earliest SDK version that supports the system, and click **Next**. The activity effect presentation screen is displayed.



Step 4 Select **Blank Activity** and click **Finish** to finish project creation.

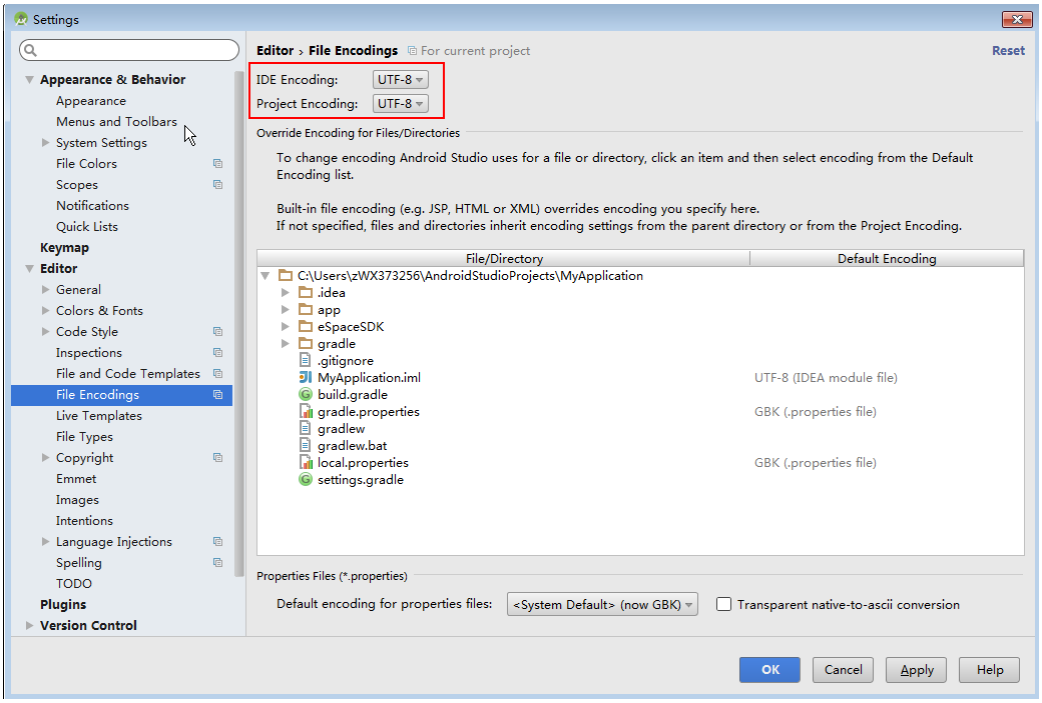


----End

4.4 Setting the Encoding Format

In the Android Studio, choose **File** > **Settings** > **File Encodings**, and set the encoding format.

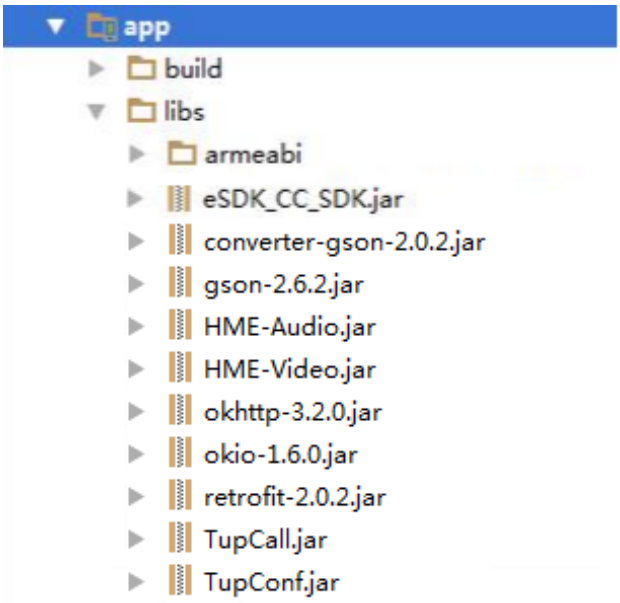
Select **UTF-8** from the **IDE Encoding** and **Project Encoding** drop-down lists respectively, and click **OK**.



4.5 Importing the Related JAR Packages

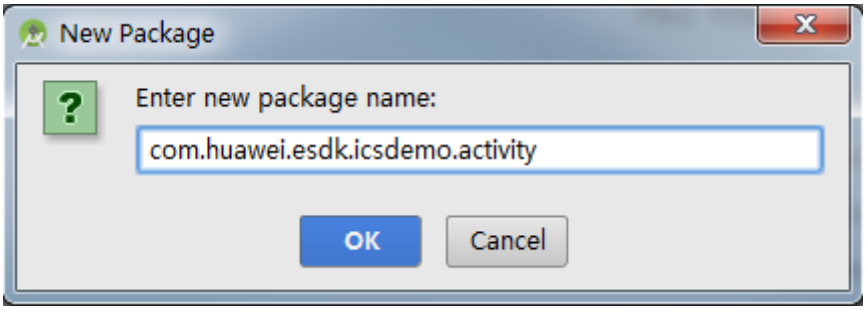
Decompress the downloaded resource **eSDK_ICP_SDK_Android_V2.1.10.zip**. After the decompression, copy the **jar** file to the **libs** folder in the project directory and copy the **so** file to the **armeabi** folder.

Click  to refresh the project. If a small arrow exists before a jar package, this package is successfully imported.

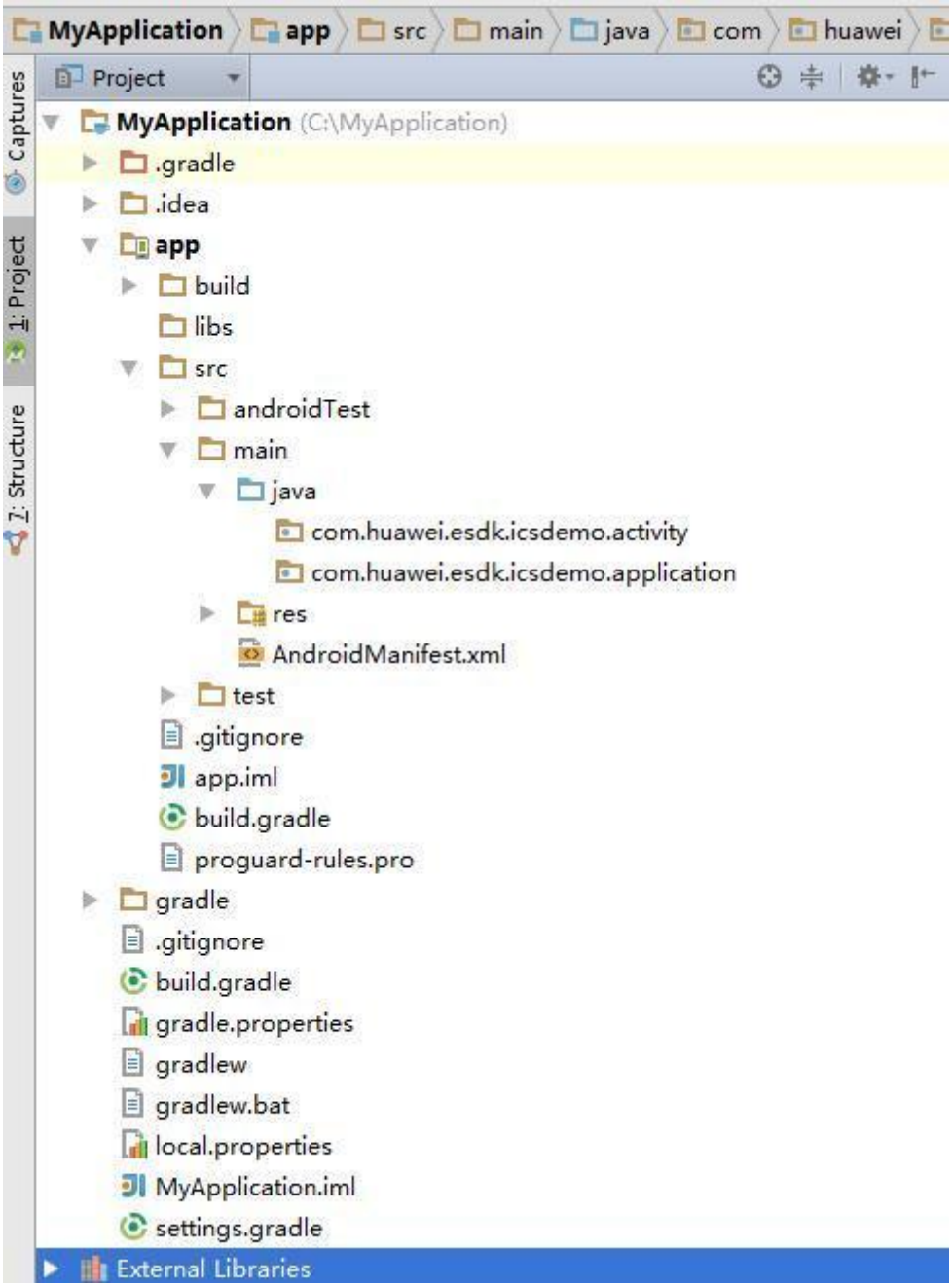


4.6 Creating a Package

Step 1 Choose **App** > **src** > **main** > **java**, right-click the **java** folder, and choose **New** > **Package**. The **New Package** window is displayed. Type **com.huawei.esdk.icsdemo.activity** in **Enter new package name**.



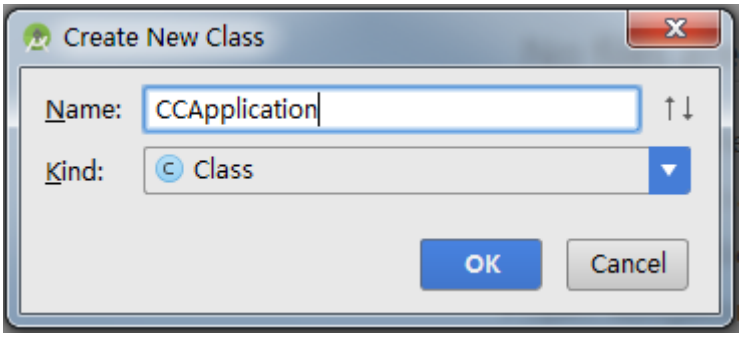
Step 2 Create a package named **com.huawei.esdk.icsdemo.application** in the method described in Step 1.



----End

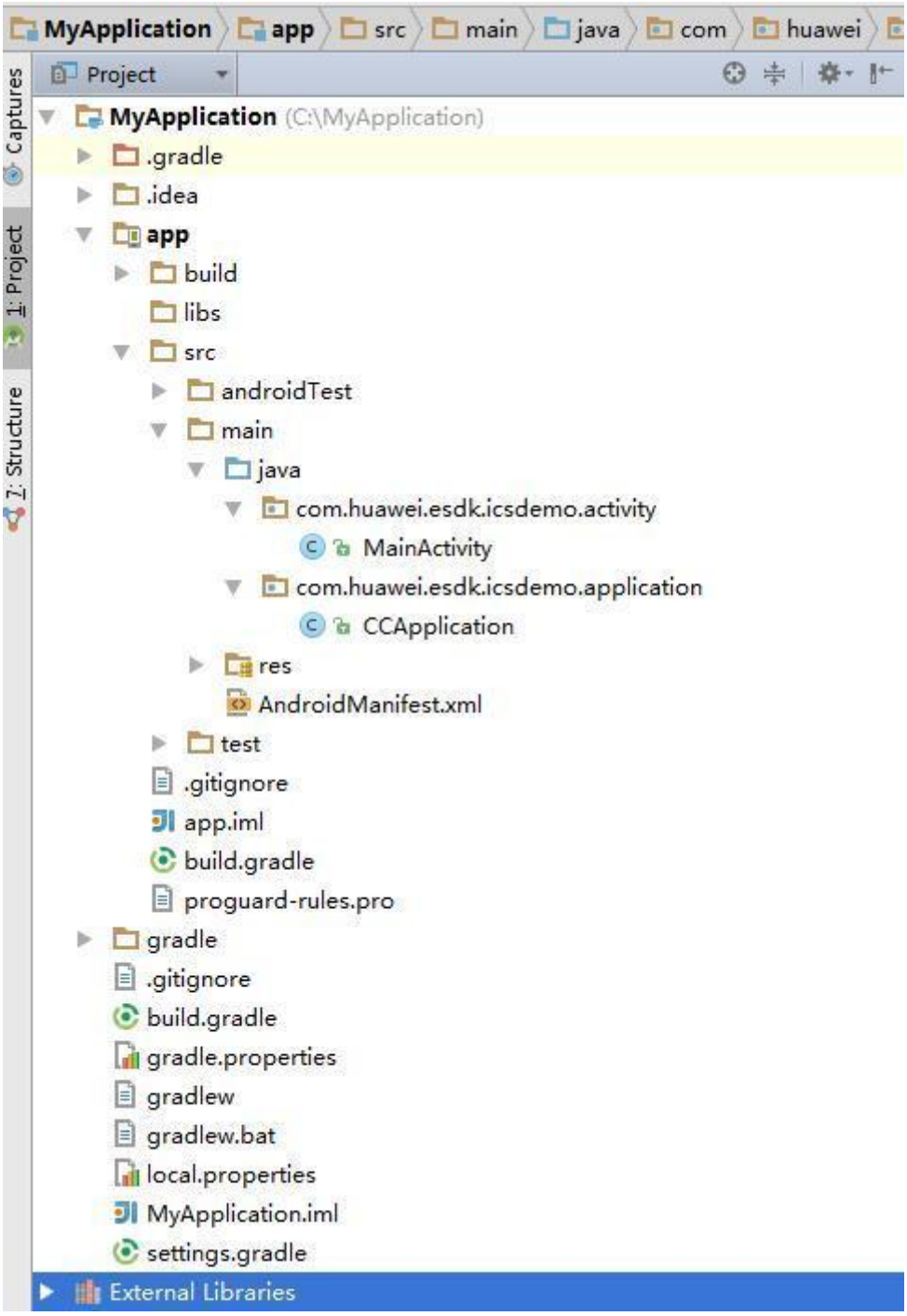
4.7 Creating a Class

Step 1 Right-click the **com.huawei.esdk.icsdemo.application** package and choose **New**. The **Java Class** window is displayed.



Step 2 Enter **CCApplication** in **Name**, and click **OK** to finish class creation.

Step 3 Repeat the preceding steps, and create **MainActivity** in the **com.huawei.esdk.icsdemo.activity** package, as shown in the following figure. For the project already having this file, change **MainActivity** and fill corresponding codes. For details, see the 8 Appendix.



Step 4 In the **res** directory, create the **values-zh** folder, create **strings.xml**, and fill corresponding codes.

----End

4.8 Implementing the Code

Overall Structure

```
CC Demo
-- src
--com.huawei.esdk.icsdemo.activity
--MainActivity.java
--com.huawei.esdk.icsdemo.application
--CCApplication.java
--res
--layout
--activity main.xml
--values
--strings.xml
--values-zh
--strings.xml
--AndroidManifest.xml
--build.gradle
```

Source code link: See the 8.1 Hello World Source Code File.

Key Code

The following provides the references for some key codes:

1. MainActivity

```
//Initialize view, monitor, and filter
//java code
private void initView()
{
etIP = (EditText) findViewById(R.id.et ip);
etPort = (EditText) findViewById(R.id.et port);
etName = (EditText) findViewById(R.id.et name);
btnLogin = (Button) findViewById(R.id.btn login);

filter = new IntentFilter();
filter.addAction(NotifyMessage.AUTH MSG ON LOGIN);
filter.addAction(NotifyMessage.AUTH MSG ON LOGOUT);
}

//Register broadcasting
@Override
protected void onResume()
{
super.onResume();
registerReceiver(receiver, filter);
}

@Override
protected void onPause()
{
super.onPause();
unregisterReceiver(receiver);
}
```

```
//receiver logic

private BroadcastReceiver receiver = new BroadcastReceiver()
{
@Override
public void onReceive(Context context, Intent intent)
{
String action = intent.getAction();
BroadMsg broadMsg = (BroadMsg) intent
.getSerializableExtra(NotifyMessage.CC_MSG_CONTENT);

if (NotifyMessage.AUTH_MSG_ON_LOGIN.equals(action))
{
//When retcode is blank, the error code is used to prompt the user.
if (null == broadMsg.getRetCode())
{
Toast.makeText(MainActivity.this, getString(R.string.login_fail) +
broadMsg.getErrorCode(), Toast.LENGTH_SHORT).show();
}
//when retcode is not blank, the error code returned from the server is used to prompt
the user.
else
{
if (("0").equals(broadMsg.getRetCode()))
{
//Login is successful.
Toast.makeText(MainActivity.this, getString(R.string.login success),
Toast.LENGTH SHORT).show();
}
else
{
Toast.makeText(MainActivity.this, "Login failed!" + broadMsg.getRetCode(),
Toast.LENGTH SHORT).show();
}
}
}
else if (NotifyMessage.AUTH MSG ON LOGOUT.equals(action))
{
if (null == broadMsg.getRetCode())
{
Toast.makeText(MainActivity.this, getString(R.string.logout fail) +
broadMsg.getErrorCode(), Toast.LENGTH SHORT).show();
}
else
{
//Logout is successful.
if (("0").equals(broadMsg.getRetCode()))
{
Toast.makeText(MainActivity.this, "Logout is
successful!" ,Toast.LENGTH SHORT).show();
}
else
{
Toast.makeText(MainActivity.this, getString(R.string.logout fail) +
broadMsg.getRetCode(), Toast.LENGTH_SHORT).show();
```

```
}
}
}
}
};
```

2. CCApplication

```
//java code
public class CCApplication extends Application
{
@Override
public void onCreate()
{
super.onCreate();
MobileCC.getInstance().setLog("eSDK", 3);
MobileCC.getInstance().initSDK(this);

}

@Override
public void onTerminate()
{
super.onTerminate();
MobileCC.getInstance().unInitSDK();
}
}
```

3. AndroidManifest

   The SDK version in the configuration file must be consistent with that in the current compiling environment.

```
//xml code
<?xml version="1.0" encoding="utf-8"?>
<manifest package="com.huawei.esdk.icsdemo"
xmlns:android="http://schemas.android.com/apk/res/android"
android:versionCode="1"
android:versionName="1.5.70">

<uses-sdk
android:minSdkVersion="15"
android:targetSdkVersion="19"/>

<uses-permission
android:name="android.permission.ACCESS WIFI STATE"></uses-permission>
<uses-permission
android:name="android.permission.RECORD AUDIO"></uses-permission>
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-permission
android:name="android.permission.ACCESS NETWORK STATE"></uses-permission>
<uses-permission
android:name="android.permission.MODIFY AUDIO SETTINGS"></uses-permission>
<uses-permission android:name="android.permission.VIBRATE"></uses-permission>
<uses-permission android:name="android.permission.GET TASKS"></uses-permission>
<uses-permission
android:name="android.permission.WRITE EXTERNAL STORAGE"></uses-permission>

<uses-permission
```

```
android:name="android.permission.RESTART_PACKAGES"></uses-permission>
<uses-permission android:name="android.permission.WAKE_LOCK"></uses-permission>
<uses-permission
android:name="android.permission.BROADCAST_STICKY"></uses-permission>
<uses-permission android:name="android.permission.BLUETOOTH"></uses-permission>
<uses-permission
android:name="android.permission.READ_PHONE_STATE"></uses-permission>
<uses-permission
android:name="android.permission.PROCESS_OUTGOING_CALLS"></uses-permission>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission
android:name="android.permission.CHANGE_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.CAMERA"></uses-permission>

<uses-feature
android:name="android.hardware.camera"
android:required="true"/>

<application
android:name="com.huawei.esdk.icsdemo.application.CCApplication"
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:hardwareAccelerated="false"
android:theme="@style/AppTheme">
<activity
android:name=".activity.MainActivity"
android:label="@string/app name"
android:theme="@style/AppTheme">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>

<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<service android:name="com.huawei.AudioDeviceAndroidService"/>
</application>

</manifest>
```

4.9 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 commission and run the application.

Without Huawei CC Environment

If you have not deployed Huawei ICP solution, log in to the , apply for the ICP environment for free, and commission and run the application.

Commissioning and Running the Application

Step 1 Click the green arrow  in the IDE toolbar to start the application.

Step 2 After starting the application, set network parameters, and click **Login**. The message "Login success" is displayed.

Step 3 After the login is successful, click **Logout**. The message "Logout success!" is displayed.

----End

5 Typical Development Scenarios

5.1 Scenario 1: Login and Logout

Function Description

The REST interface provided by the eSDK service end is used for login and logout.

Sample Code

```
//java code
//Set filter
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.AUTH MSG ON LOGIN);
filter.addAction(NotifyMessage.AUTH MSG ON LOGOUT);

private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);

//Initialize receiver
private BroadcastReceiver receiver = new BroadcastReceiver()
{
@Override
public void onReceive(Context context, Intent intent)
{
String action = intent.getAction();
BroadMsg broadMsg = (BroadMsg) intent
.getSerializableExtra(NotifyMessage.CC MSG CONTENT);

if (NotifyMessage.AUTH MSG ON LOGIN.equals(action))
{
```

```
if (("0").equals(broadMsg.getRecode()))
{
//Login is successful.
}
}
else if(NotifyMessage.AUTH_MSG_ON_LOGOUT.equals(action))
{
if (("0").equals(broadMsg.getRecode()))
{
//Logout is successful.
}
else
{
//Logout failed.
}
}
}
};

//Monitor broadcasting
registerReceiver(receiver, filter);

//Login operation
if (0 != MobileCC.getInstance().login("1", etName.getText()
.toString().trim()))
{
//Login request is already sent.
}
//Logout operation
MobileCC.getInstance().logout();
```

Sample Screen

Figure 5-1 Screen before login

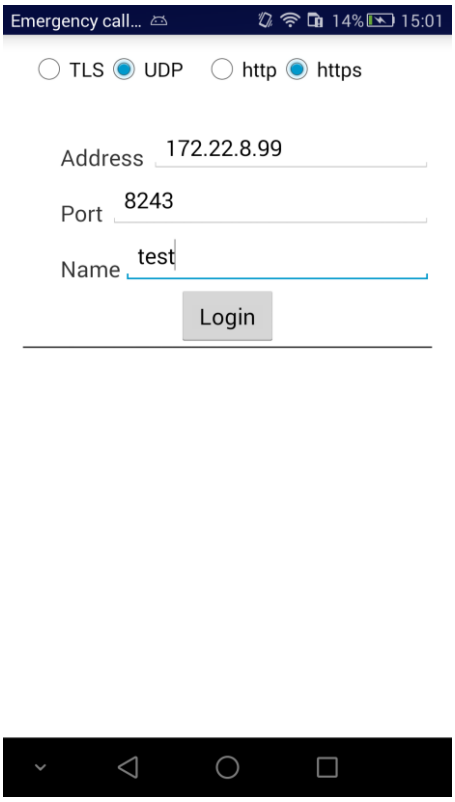


Figure 5-2 Screen after login

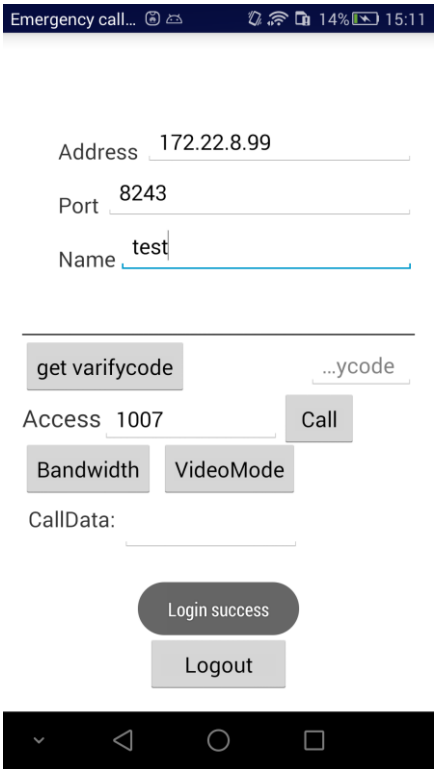
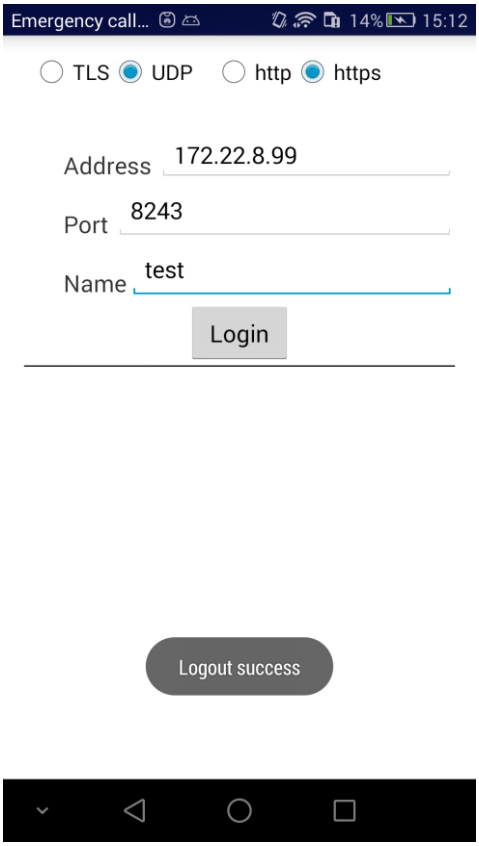


Figure 5-3 Screen after logout



5.2 Scenario 2: TP Call Function

Function Description

Initiate a call: A user initiates a call.

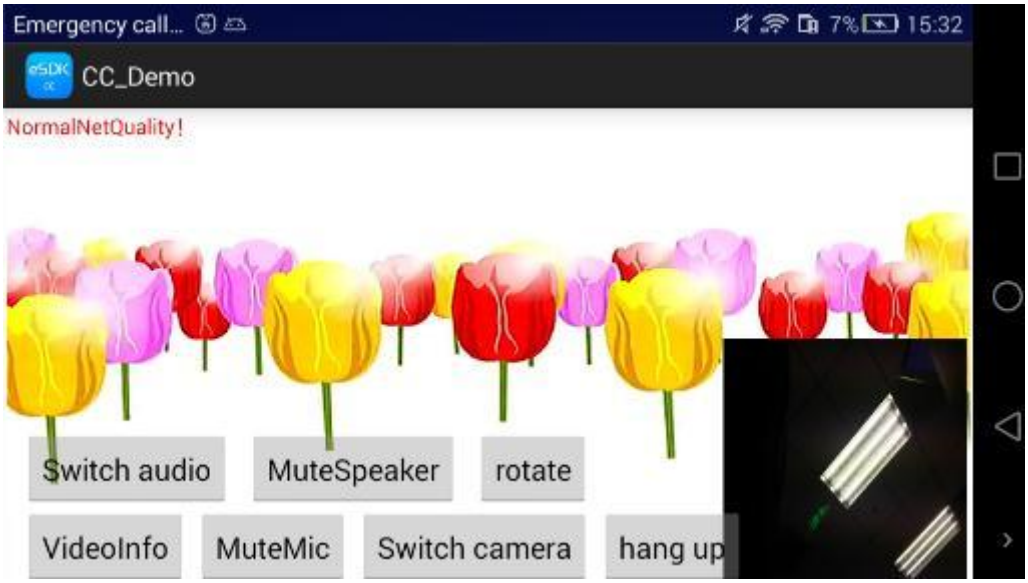
End a call: A user ends the current call.

Sample Code

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//Set filter
private IntentFilter filter;
filter = new IntentFilter();
filter.addAction(NotifyMessage.CALL_MSG_ON_CONNECTED);
filter.addAction(NotifyMessage.CALL_MSG_REFRESH_LOCALVIEW);
filter.addAction(NotifyMessage.CALL_MSG_REFRESH_REMOTEVIEW);
filter.addAction(NotifyMessage.CALL_MSG_ON_DISCONNECTED);

//Initialize receiver
private BroadcastReceiver receiver = new BroadcastReceiver()
```

```
{
@Override
public void onReceive(Context context, Intent intent)
{
String action = intent.getAction();
BroadMsg broadMsg = (BroadMsg) intent
.getSerializableExtra(NotifyMessage.CC_MSG_CONTENT);

if (NotifyMessage.CALL_MSG_ON_CONNECTED.equals(action))
{
//Connected to the agent
}
else if (NotifyMessage.CALL_MSG_REFRESH_LOCALVIEW.equals(action))
{
//Refresh the local video screen
MobileCC.getInstance().setVideoContainer(ChatActivity.this, localView, null);
}
else if (NotifyMessage.CALL_MSG_REFRESH_REMOTEVIEW.equals(action))
{
//Refresh the remote video screen
MobileCC.getInstance().setVideoContainer(ChatActivity.this, null, remoteView);
}
else if (NotifyMessage.CALL_MSG_ON_DISCONNECTED.equals(action))
{
//Disconnected from the agent, and the call the released.
}
}
}

//Register receiver

@Override
protected void onResume()
{
super.onResume();
registerReceiver(receiver, filter);
}

//Call operation
MobileCC.getInstance().makeCall("8888", MobileCC.SERVER TP + "", "data", verfiCode);

//Release the call
MobileCC.getInstance().releaseCall();
```

Sample Screen



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

- MS to send text code

```
//java code
MobileCC.getInstance().sendMsg("Help!Help!");
```

- MS voice call

```
//java code
MobileCC.getInstance().makeCall(audioAccessCode, MobileCC.AUDIO CALL, data,
etVerifycode.getText().toString());
```

- MS in the video call code

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//Set filter
private IntentFilter filter;
```

```
filter = new IntentFilter();
filter.addAction(NotifyMessage.CALL_MSG_ON_CONNECTED);
filter.addAction(NotifyMessage.CALL_MSG_ON_DISCONNECTED);
filter.addAction(NotifyMessage.CALL_MSG_ON_APPLY_MEETING);
filter.addAction(NotifyMessage.CALL_MSG_USER_STATUS);
filter.addAction(NotifyMessage.CALL_MSG_ON_STOP_MEETING);



//Initialize receiver
 private BroadcastReceiver receiver = new BroadcastReceiver()
    {
       @Override
       public void onReceive(Context context, Intent intent)
       {
          String action = intent.getAction();
          BroadMsg broadMsg = (BroadMsg) intent
             .getSerializableExtra(NotifyMessage.CC_MSG_CONTENT);

          if (NotifyMessage.CALL_MSG_ON_CONNECTED.equals(action))
          {
             //Connect with the agent
          }
          else if (NotifyMessage.CALL MSG USER STATUS.equals(action))
          {
             //The meeting was created successfully
             MobileCC.getInstance().setVideoContainer(MeetingActivity.this,
mLlLocalSurface, mLlRemoteSurface);

          }
else if (NotifyMessage.CALL MSG ON APPLY MEETING.equals(action))
          {

             if (null == broadMsg.getRetCode())
             {
                Toast.makeText(MSChatActivity.this, "Fail to apply for a meeting, the
error code is: " + broadMsg.getErrorCode(), Toast.LENGTH_SHORT).show();
             }
             else
             {
                String retcode = broadMsg.getRetCode();

                if (MobileCC.MESSAGE OK.equals(retcode))
                {
                   Toast.makeText(MSChatActivity.this, "Apply for meeting
successful", Toast.LENGTH SHORT).show();
                }
                else
                {
                   Toast.makeText(MSChatActivity.this, "Fail to apply for a meeting,
the error code is: " + retcode, Toast.LENGTH SHORT).show();
                }
             }
          }
```

```
          else if (NotifyMessage.CALL_MSG_ON_DISCONNECTED.equals(action))
          {
                 //Disconnect from the agent, the call is released          }
          else if (NotifyMessage.CALL_MSG_ON_STOP_MEETING.equals(action))
          {
              //Quit the meeting
          }
    }
    }


//Register receiver
   @Override
    protected void onResume()
    {
       super.onResume();
       localBroadcastManager.registerReceiver(receiver, filter);    }

//Screen conference call operation
MobileCC.getInstance().makeCall("60021", MobileCC.VIDEO_CALL, data. " ");

//Release the call
MobileCC.getInstance().releaseCall();
```

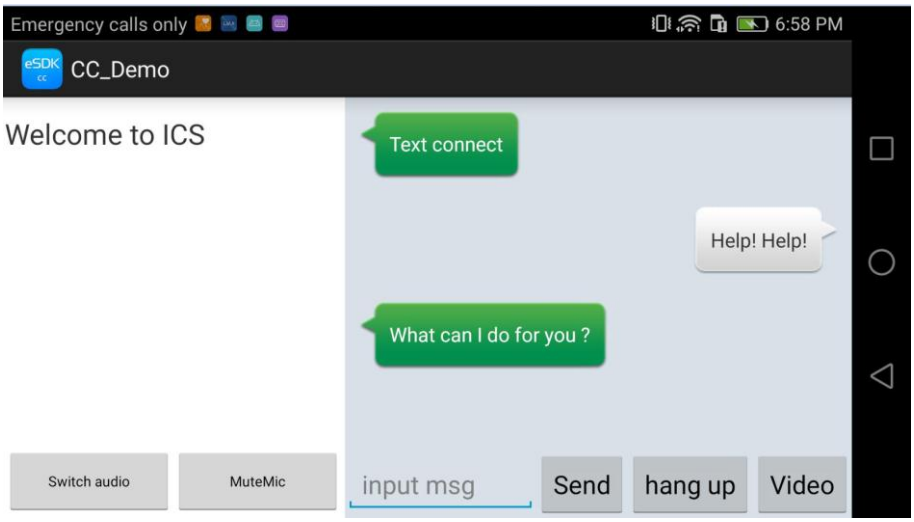
- MS in the desktop to share the code

```
//java code
private LocalBroadcastManager localBroadcastManager =
LocalBroadcastManager.getInstance(this);
//set filter
IntentFilter filter = new IntentFilter();
filter.addAction(NotifyMessage.CALL MSG USER RECEIVE SHARED DATA);

//Initializereceiver

 private BroadcastReceiver receiver = new BroadcastReceiver()
    {
       @Override
       public void onReceive(Context context, Intent intent)
       {
          String action = intent.getAction();
          BroadMsg broadMsg = (BroadMsg) intent
              .getSerializableExtra(NotifyMessage.CC MSG CONTENT);

          if (NotifyMessage. CALL MSG USER RECEIVE SHARED DATA.equals(action))
          {
              //Received the shared notification
          }

        }
    }

//Register receiver
   @Override
    protected void onResume()
    {
       super.onResume();
```

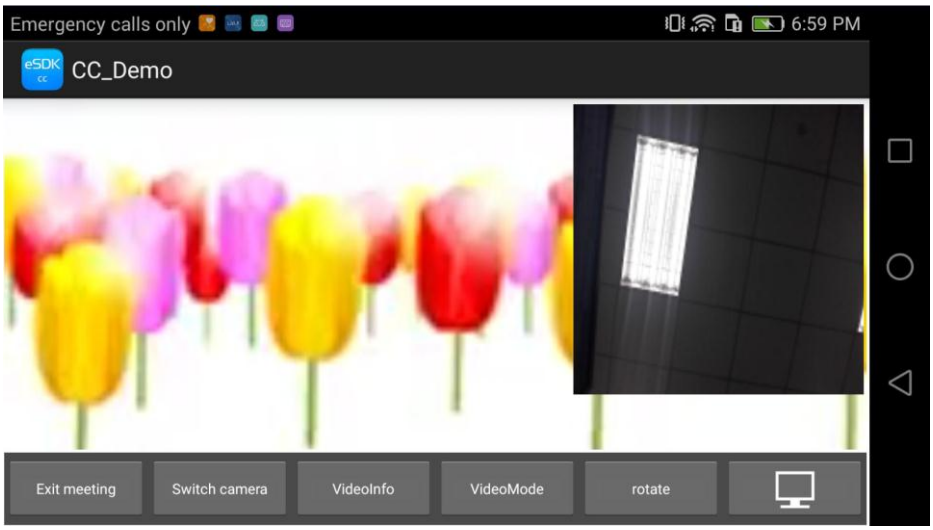
```
        localBroadcastManager.registerReceiver(receiver, filter);
    }

//Load the shared screen
MobileCC.getInstance().setDesktopShareContainer(getBaseContext(),
        desktopSharedLayout); // Receive the shared data, set the display container
```

Sample Screen

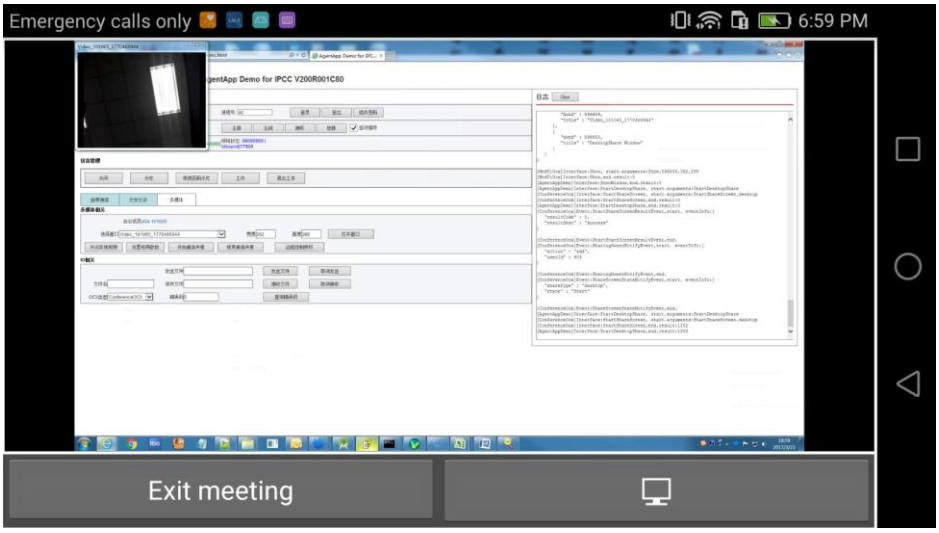
- MS text links and voice calls



- MS video interface



- MS in the desktop sharing interface



5.4 Scenario 3: Device Control

Function Description

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

Sample Code

- Switching the camera

```
//java code
MobileCC.getInstance().switchCamera();
```

- Switching between the speaker mode and earpiece mode

```
//java code
//Speaker mode
MobileCC.getInstance().changeAudioRoute(MobileCC.getInstance().AUDIO ROUTE SPEA
KER);

//Earpiece mode
MobileCC.getInstance().changeAudioRoute(MobileCC.getInstance().AUDIO ROUTE RECE
IVER);
```

- Muting the microphone

```
//java code
//Mute the microphone
MobileCC.getInstance().setMicMute(true);

//Cancel muting the microphone
MobileCC.getInstance().setMicMute(false);
```

- Muting the speaker

```
//java code
//Mute the speaker
MobileCC.getInstance().setSpeakerMute(true);
```

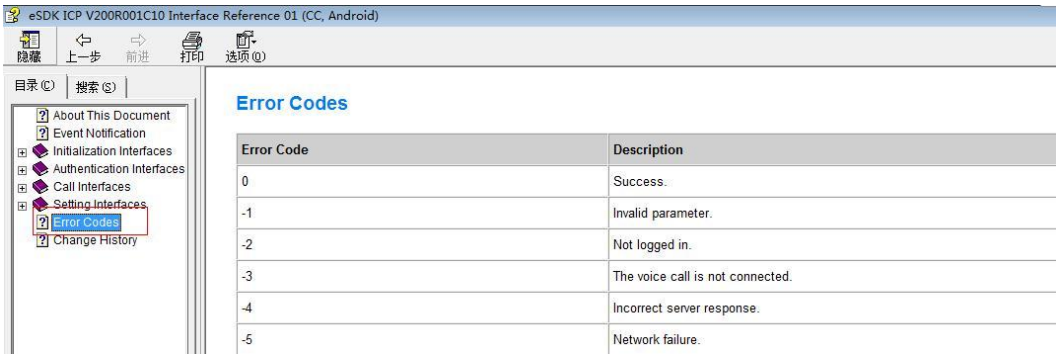
```
//Cancel muting the speaker
MobileCC.getInstance().setSpeakerMute(false);
```

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

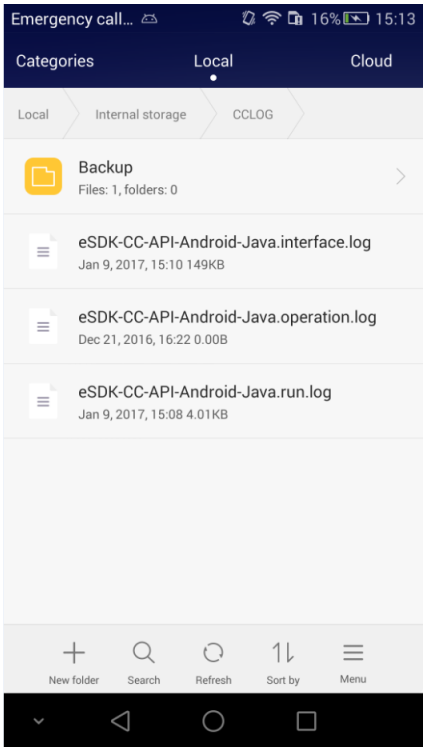
Returned upon interface invocation

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.

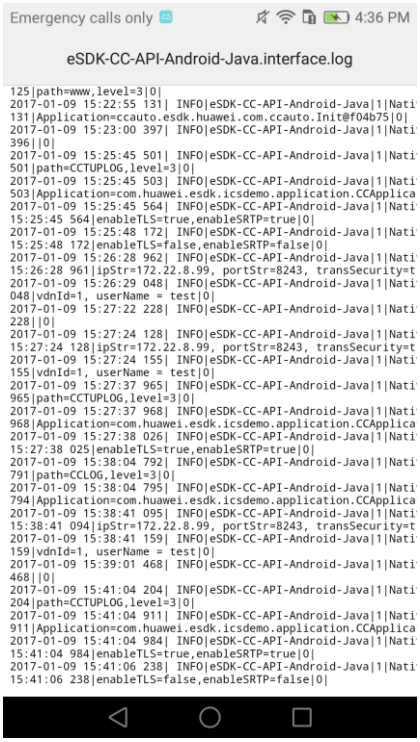
Obtained from the log files

You can obtain the eSDK interface return value from the interface log file.

By default, the log file is generated in **/sdcard/CCLOG/eSDK-CC-API-Android.interface.log.**



The interface log file records interface invocation results in detail.



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-Android|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-Android|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	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 CCApplication.java

```
//java code
package com.huawei.esdk.icsdemo.application;
import android.app.Application;
import com.huawei.esdk.cc.MobileCC;
public class CCApplication extends Application
{

@Override
public void onCreate()
{
super.onCreate();
MobileCC.getInstance().setLog("CCLOG", 3);
MobileCC.getInstance().initSDK(this);

}

@Override
public void onTerminate()
{
super.onTerminate();
MobileCC.getInstance().unInitSDK(); // Stop the SDK service
}
}
```

8.1.2 MainActivity.java

```
//java code
package com.huawei.esdk.icsdemo.activity;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
```

```
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;

import android.widget.Toast;

import com.huawei.esdk.cc.MobileCC;
import com.huawei.esdk.cc.common.BroadMsg;
import com.huawei.esdk.cc.common.NotifyMessage;
import com.huawei.esdk.icsdemo.R;

public class MainActivity extends Activity implements View.OnClickListener
{
private EditText etIP;
private EditText etPort;
private EditText etName;
private Button btnLogin;
private Button btnExit;
private IntentFilter filter;

@Override
protected void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.activity main);
initView();
}

/**
* Initialize view, monitor, and filter
*/
private void initView()
{
etIP = (EditText) findViewById(R.id.et ip);
etPort = (EditText) findViewById(R.id.et port);
etName = (EditText) findViewById(R.id.et name);
btnLogin = (Button) findViewById(R.id.btn login);
btnExit = (Button) findViewById(R.id.btn exit);

btnLogin.setOnClickListener(this);
btnExit.setOnClickListener(this);

filter = new IntentFilter();
filter.addAction(NotifyMessage.AUTH MSG ON LOGIN);
filter.addAction(NotifyMessage.AUTH MSG ON LOGOUT);
}

@Override
protected void onResume()
{
```

```
super.onResume();
registerReceiver(receiver, filter);
}

@Override
public void onClick(View view)
{
switch (view.getId())
{
case R.id.btn_login:
login();
break;

case R.id.btn_exit:
MobileCC.getInstance().logout();
break;

default:
break;
}
}

private BroadcastReceiver receiver = new BroadcastReceiver()
{
@Override
public void onReceive(Context context, Intent intent)
{
String action = intent.getAction();
BroadMsg broadMsg = (BroadMsg) intent
.getSerializableExtra(NotifyMessage.CC MSG CONTENT);

if (NotifyMessage.AUTH MSG ON LOGIN.equals(action))
{
//When retcode is blank, the error code is used to prompt the user.
if (null == broadMsg.getRetCode())
{
Toast.makeText(MainActivity.this, getString(R.string.login fail) +
broadMsg.getErrorCode(), Toast.LENGTH SHORT).show();
}
//when retcode is not blank, the error code returned from the server is used to prompt
the user.
else
{
if (("0").equals(broadMsg.getRetCode()))
{
//Login is successful.
Toast.makeText(MainActivity.this, getString(R.string.login success),
Toast.LENGTH SHORT).show();
}
else
{
Toast.makeText(MainActivity.this, "Login failed!" + broadMsg.getRetCode(),
Toast.LENGTH SHORT).show();
}
}
```

```
}
else if (NotifyMessage.AUTH_MSG_ON_LOGOUT.equals(action))
{
if (null == broadMsg.getRetCode())
{
Toast.makeText(MainActivity.this, getString(R.string.logout_fail) +
broadMsg.getErrorCode(), Toast.LENGTH_SHORT).show();
}
else
{
//Logout is successful.
if (("0").equals(broadMsg.getRetCode()))
{
Toast.makeText(MainActivity.this, "Logout is successful!" ,Toast.LENGTH_SHORT).show();
}
else
{
Toast.makeText(MainActivity.this, getString(R.string.logout_fail) +
broadMsg.getRetCode(), Toast.LENGTH_SHORT).show();
}
}
}
}
};

private int setHostAddress()
{
String ipStr = etIP.getText().toString();
String portStr = etPort.getText().toString();
return MobileCC.getInstance().setHostAddress(ipStr, portStr, false,
MobileCC.SERVER TP);
}

private void login()
{
//Fold the keypad
((InputMethodManager)
getSystemService(INPUT METHOD SERVICE)).hideSoftInputFromWindow(MainActivity.this.
getCurrentFocus().getWindowToken(), InputMethodManager.HIDE NOT ALWAYS);
if (0 == (setHostAddress()))
{
if (0 != MobileCC.getInstance().login("1", etName.getText().toString().trim()))
{
Toast.makeText(MainActivity.this, "Invalid user name, please input again",
Toast.LENGTH SHORT).show();
}
}
else
{
Toast.makeText(MainActivity.this, "Incorrect network parameter",
Toast.LENGTH SHORT).show();
}
}
@Override
protected void onDestroy()
```

```
{
unregisterReceiver(receiver);
android.os.Process.killProcess(android.os.Process.myPid());
super.onDestroy();
}
}
```

8.1.3 build.gradle

This file and **AndroidMainifest.xml** must be changed based on the current compiling environment.

```
apply plugin: 'com.android.application'

dependencies {
compile fileTree(include: ['*.jar'], dir: 'libs')
compile files('libs/android-support-v4.jar')
}

android {
compileSdkVersion 19
buildToolsVersion '21.1.2'

buildTypes {
release {
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
}
sourceSets {
main {
jniLibs.srcDirs = ['libs']
}
}
}
```

8.1.4 AndroidMainifest.xml

```
//xml code
<?xml version="1.0" encoding="utf-8"?>
<manifest package="com.huawei.esdk.icsdemo"
xmlns:android="http://schemas.android.com/apk/res/android"
android:versionCode="1"
android:versionName="1.5.70">

<uses-sdk
android:minSdkVersion="15"
android:targetSdkVersion="19"/>

<uses-permission
android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.RECORD_AUDIO"></uses-permission>
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
<uses-permission
```

```
android:name="android.permission.MODIFY_AUDIO_SETTINGS"></uses-permission>
<uses-permission android:name="android.permission.VIBRATE"></uses-permission>
<uses-permission android:name="android.permission.GET_TASKS"></uses-permission>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>

<uses-permission
android:name="android.permission.RESTART_PACKAGES"></uses-permission>
<uses-permission android:name="android.permission.WAKE_LOCK"></uses-permission>
<uses-permission
android:name="android.permission.BROADCAST_STICKY"></uses-permission>
<uses-permission android:name="android.permission.BLUETOOTH"></uses-permission>
<uses-permission
android:name="android.permission.READ_PHONE_STATE"></uses-permission>
<uses-permission
android:name="android.permission.PROCESS_OUTGOING_CALLS"></uses-permission>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission
android:name="android.permission.CHANGE_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.CAMERA"></uses-permission>

<uses-feature
android:name="android.hardware.camera"
android:required="true"/>

<application
android:name="com.huawei.esdk.icsdemo.application.CCApplication"
android:allowBackup="true"
android:icon="@mipmap/ic launcher"
android:label="@string/app name"
android:hardwareAccelerated="false"
android:theme="@style/AppTheme">
<activity
android:name=".activity.MainActivity"
android:label="@string/app name"
android:theme="@style/AppTheme">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>

<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<service android:name="com.huawei.AudioDeviceAndroidService"/>
</application>

</manifest>
```

8.1.5 values/strings.xml

```
//xml code
<?xml version="1.0" encoding="utf-8"?>
<resources>

<string name="app name">CC Demo</string>
<string name="hello_world">Hello world!</string>
```

```
<string name="action_settings">Settings</string>

<string name="login_fail">Login fail! errorCode is:</string>
<string name="login_success">Login success</string>
<string name="logout_fail">Logout fail! errorCode is:</string>

<string name="address">Address</string>
<string name="port">Port</string>
<string name="name">Name</string>
<string name="login">Login</string>
<string name="logout">Logout</string>

</resources>
```

8.1.6 values-zh/strings.xml

```
//xml code
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app name">CC Demo</string>
<string name="hello world">Hello world!</string>
<string name="action settings">Settings</string>

<string name="login fail">Login failed. The error code is: </string>
<string name="login success">Login is successful</string>
<string name="logout fail">Logout failed. The error code is: </string>

<string name="address">Address</string>
<string name="port">Port</string>
<string name="name">User name</string>
<string name="login">Login</string>
<string name="logout">Logout</string>

</resources>
```

8.1.7 layout/activity_main.xml

```
//xml code
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout height="match parent"
android:paddingBottom="@dimen/activity vertical margin"
android:paddingLeft="@dimen/activity horizontal margin"
android:paddingRight="@dimen/activity horizontal margin"
android:paddingTop="@dimen/activity vertical margin"
tools:context=".activity.MainActivity">

<TextView
android:id="@+id/tv ip"
android:layout width="wrap content"
android:layout height="wrap content"
android:paddingLeft="30dp"
android:paddingTop="70dp"
```

```
android:text="@string/address"
android:textSize="26dp"/>

<EditText
android:id="@+id/et_ip"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignBottom="@+id/tv_ip"
android:layout_marginLeft="5dp"
android:layout_toRightOf="@+id/tv_ip"
android:singleLine="true"
android:ellipsize="start"
android:text="172.22.9.40"

android:textSize="14dp"/>
<!-- TP: 172.22.9.40 MS: 10.174.5.58 -->
<TextView
android:id="@+id/tv_port"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/tv_ip"
android:layout_marginTop="20dp"
android:paddingLeft="30dp"
android:text="@string/port"
android:textSize="26dp"/>

<EditText
android:id="@+id/et_port"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignBottom="@+id/tv_port"
android:layout_below="@+id/tv_ip"
android:layout_marginLeft="5dp"
android:layout_toRightOf="@+id/tv_port"
android:singleLine="true"
android:ellipsize="start"
android:text="8280"
android:textSize="14dp"/>

<TextView
android:id="@+id/tv_name"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/tv_port"
android:layout_marginTop="20dp"
android:paddingLeft="30dp"
android:text="@string/name"
android:textSize="26dp"/>

<EditText
android:id="@+id/et_name"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignBottom="@+id/tv_name"
android:layout_below="@+id/tv_port"
```

```
android:layout_marginLeft="2dp"
android:layout_toRightOf="@+id/tv_name"
android:editable="true"
android:singleLine="true"
android:ellipsize="start"
android:textSize="14dp"/>

<Button
android:id="@+id/btn_login"
android:layout_width="120dp"
android:layout_height="wrap_content"
android:layout_below="@+id/tv_name"
android:layout_centerHorizontal="true"
android:layout_marginTop="15dp"
android:text="@string/login"/>

<Button
android:id="@+id/btn_exit"
android:layout_width="120dp"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true"
android:text="@string/logout"/>

</RelativeLayout>
```