eSDK CloudVC 20.1.RC1 开发指南(Android)

eSDK CloudVC 20.1.RC1 开发指南 (Android)

文档版本 01

发布日期 2020-03-06

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

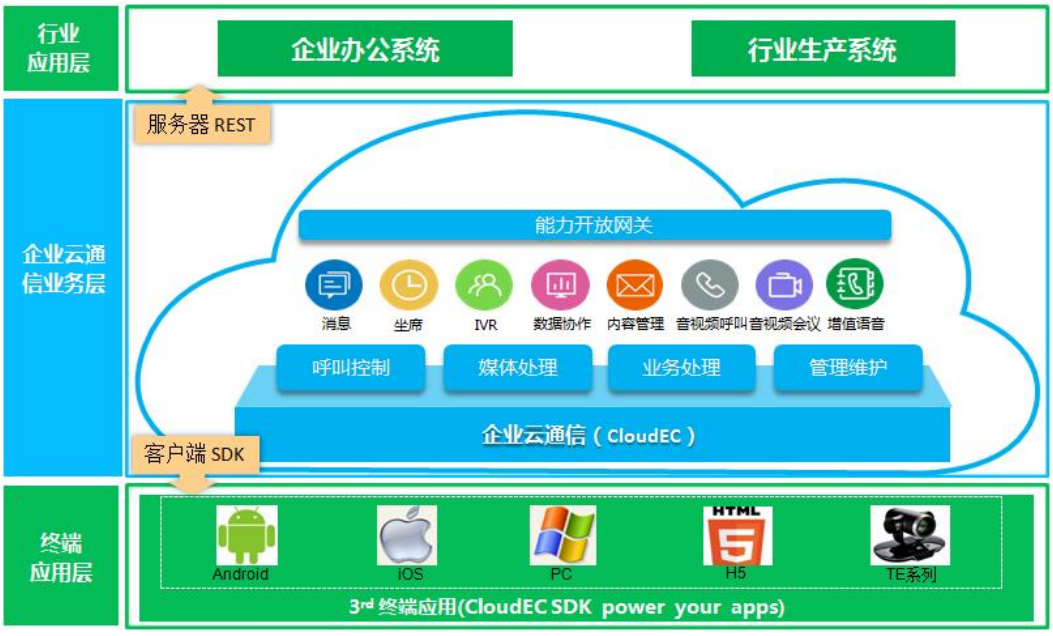
eSDK CloudVC 20.1.RC1 开发指南(Android)

目 录

1 eSDK CloudVC 是什么?

CloudVC 简介

Cloud VC （Cloud Video Conference）是华为企业云视讯解决方案，它是面向政府、交通、安全、金融、大企业、中小企业等领域，提供高临场感的远程会议、桌面及移动视频接入、企业流媒体应用等场景下的视频会议整体解决方案。Cloud VC融合了语音、视频、数据等多种媒体能力，为企业提供一站式的融合的通信平台，包括高清音视频会议、Web会议、协作等丰富的业务，可适配不同企业不同的场景应用。视讯解决方案在架构上分为业务平台层、媒体处理层、用户接入层，各层产品功能紧密配合。



服务器REST：北向通过能力开放网关提供网络侧REST接口，支持与第三方行业系统基于HTTPS进行交互。

客户端SDK：南向开放提供Windows/Android/iOS平台的客户端SDK，第三方客户端应用通过简单的接口调用来集成相关通信能力。

eSDK CloudVC 提供什么?

华为CloudVC二次开发目前提供的Android调用方式，将华为CloudVC能力通过接口调用的方式对外开放。

我们提供的eSDK CloudVC包内容由以下几部分：

- **SDK包和文档**

  CloudVC二次开发使用的SDK包和文档，提供CloudVC接口的Android动态库、接口参考、开发指南。详情请参见**SDK和文档下载路径**。

- **Sample Codes**

  CloudVC系列的Sample Codes，演示如何调用接口，帮助您完成企业通信相关业务开发。详情请参见**Sample Codes下载路径**。

eSDK CloudVC 20.1.RC1 开发指南(Android)

2 内容导航

须知

因相应SDK版本暂仅适用于CloudVC解决方案下的融合会议组网，所以此文档中描述的功能全部仅适用于CloudVC解决方案下的融合会议组网。

本开发指南主要描述CloudVC开放的常用功能，以及指导您如何调用SDK标准化接口使用这些功能。主要内容如下：

1. **相关资源**：二次开发过程中可能涉及到的软件、文档资源链接、技术支持。包括sample codes下载链接、接口参考、介绍如何申请远程实验室和开发过程中技术支持渠道。
2. **Hello World**：如果你仅仅是尝试把SDK运行起来，您应该首先看这部分内容，它会详细说明如何下载及安装SDK以及如何配置您的开发环境。
3. **业务开发指导**：介绍TUP开放性的典型功能场景，包括开发流程、样例代码以及注意事项等。本开发指南提供以下典型场景：
   - **统一鉴权与登录**
   - **音视频呼叫**
   - **音视频会议**
4. **问题定位**：介绍开发过程中常见问题的定位方法。
5. **修订记录**：各版本开发指南更新细节。

阅读建议

- 如果想快速入门，可仅参考**Hello World**章节。
- 如果想深入了解CloudVC核心业务的二次开发，建议您参考**业务开发指导**章节。
- 使用SDK过程中遇到问题可以参考**问题定位**章节；或联系**eSDK技术支持**。

eSDK CloudVC 20.1.RC1 开发指南(Android)

3 相关资源

3.1 SDK 和文档下载路径

华为开发者社区

- 访问**华为开发者社区资源中心**，获取对应解决方案和平台的SDK和文档。

华为企业产品技术支持网站

- 访问**华为企业产品技术支持网站**，单击“**软件下载 > 企业业务公共**”，显示的页面选择“**eSDK解决方案 > eSDK VC**”，获取SDK。
- 访问**华为企业产品技术支持网站**，单击“**产品文档 > 企业业务公共**”，显示的页面选择“**eSDK VC**”，获取文档。

校验软件包完整性

步骤1 从“**软件数字签名(OpenPGP)验证工具**”下载软件校验需要的资源。（选择V600R019C10版本即可）

具体所需资源如**表3-1**所示。

表 3-1 资源说明

选项	说明
VerificationTools.zip	数字签名校验工具。
KEYS	华为软件数字签名公钥。

选项	说明
OpenPGP签名验证指南.pdf	数字签名验证指南。

步骤2 将被签名文件和对应的签名文件放置在同一个目录下。

步骤3 参见《 OpenPGP签名验证指南.pdf 》进行软件数字签名验证

----结束

3.2 Sample Codes 下载路径

访问**github**网站， 下载Sample Codes。

这里建议您使用Android Studio 2.3及以上版本编译执行Sample Codes。

如果需要下载及安装SDK，请参考 **Hello World**。

Sample Code名称	描述
CloudVC_Client_API_Demo_Android	CloudVC客户端SDK全量功能Demo，具备音视频呼叫、音视频设备管理、IPT业务、会议管理、会议接入、会议控制、桌面协同与共享和企业通讯录等基本功能。 暂仅适用于CloudVC解决方案下的融合会议组网。

3.3 免费申请远程实验室

华为 eSDK 远程实验室简介

华为eSDK 远程实验室致力于为合作伙伴提供真实的华为ICT 产品能力的远程对接联调环境，通过在线申请相应ICT 产品的测试帐号与权限，您不需要购置华为产品便可基于远程实验室针对相关产品进行二次开发，并实现远程对接测试认证。借助华为远程实验室，您可以“零”出差构建解决方案，“零”设备构建演示环境， 提高对接测试通过率，缩短产品二次开发周期。

华为eSDK远程实验室为开发者提供了7×24小时的免费云化实验室环境，提供真实的华为设备供开发者远程在线开发调试。借助远程实验室自助管理平台，开发者不需要购置华为产品便可基于远程实验室针对相关产品进行二次开发，并实现远程对接测试认证。

目前，华为远程实验室已发布Cloud Computing、Carrier Software、SDN、BYOD、Cloud EC、Agile Network、eLTE、Big Data、IoT、IES等多个生态圈的实验室环境。

相关信息请查询**华为开发者社区远程实验室**。

远程实验室有哪些优势

- 低门槛：官网注册用户即可申请使用，环境与预约时长、预约次数受限。
- 分级支持：环境按域划分，重点开发者、合作伙伴可访问特定环境并享受额外的预约环境。

eSDK CloudVC 20.1.RC1 开发指南(Android)

- 全球资源高速互联：建设以苏州远程实验室为中心的全球实验室分布格局，依托IT全球100ms高性能骨干网络和IT全球端到端应用保障能力。

如何免费使用远程实验室

请访问**远程实验室操作指导**，查阅远程实验室预约申请及接入使用方法。

3.4 技术支持渠道

华为云开发者社区提单

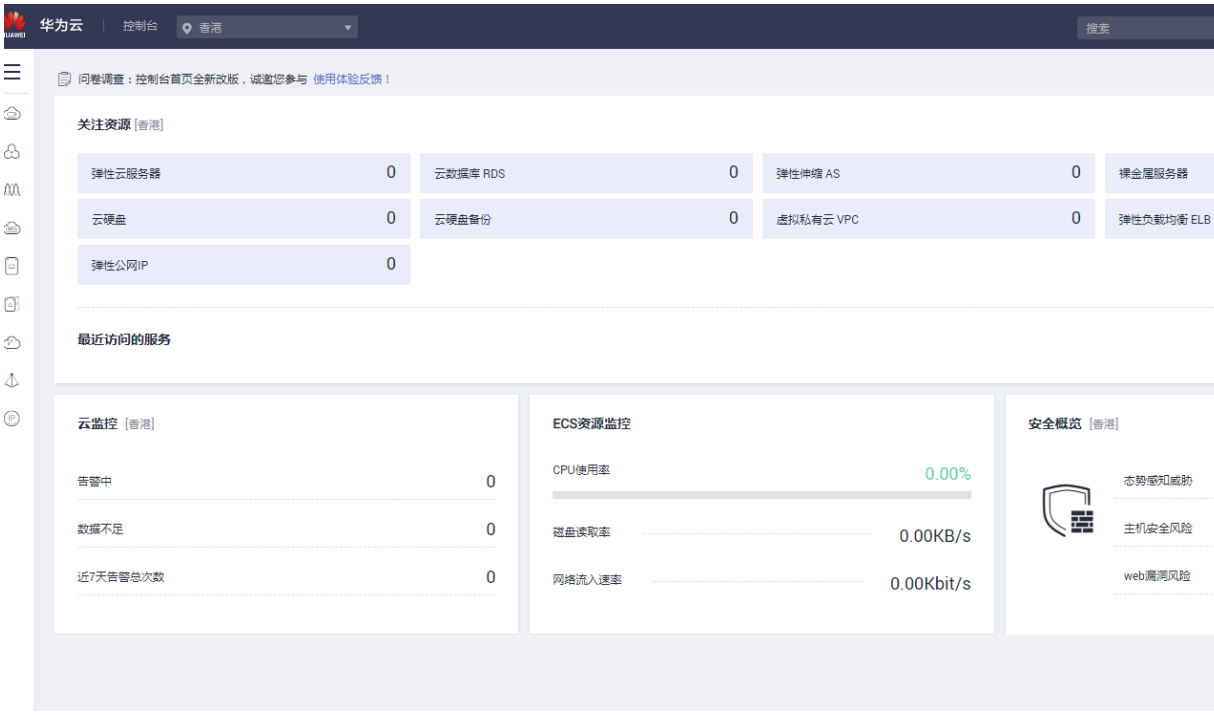
开发过程中，您有任何问题都可以在**华为云开发者社区**中提单跟踪。具体步骤如下：

步骤1 登录**华为云开发者社区**。

已注册有华为开发者社区帐号的，可直接登录。未注册的按照相关要求注册后登录。

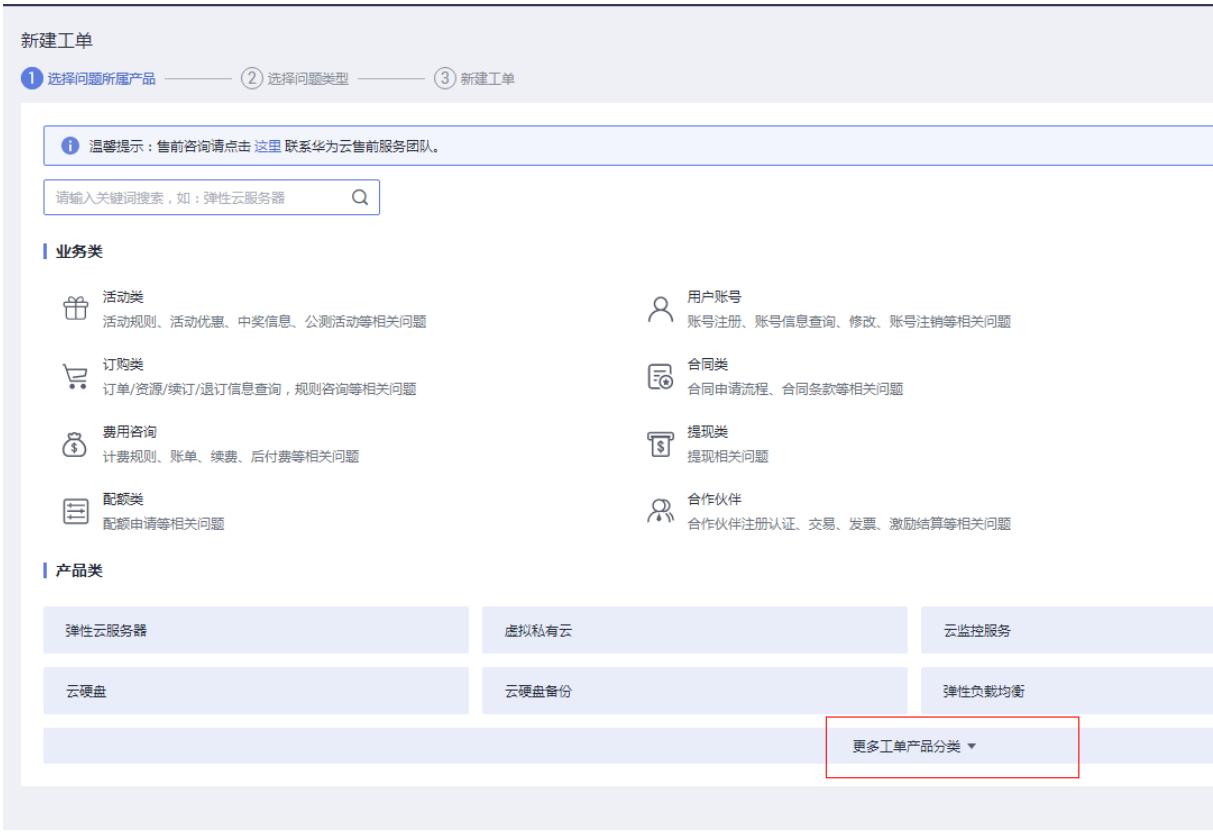
步骤2 单击“**控制台**”，进入控制台页面。

步骤3 在控制台中点击**工单，新建工单**



步骤4 在新建工单中，点击**更多工单产品分类。**

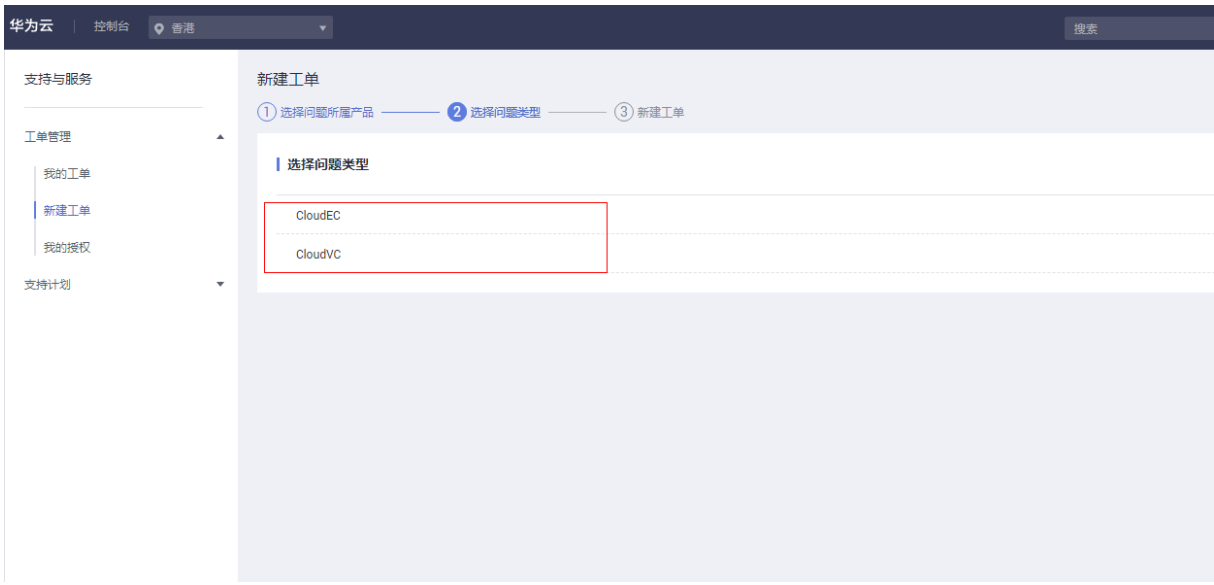
eSDK CloudVC 20.1.RC1 开发指南(Android)



在ICT开发者业务中选择**企业通信**

eSDK CloudVC 20.1.RC1 开发指南(Android)

根据产品选择**CloudEC或者CloudVC**进行提单，只有视频会议请选择CloudEC，如果有eSpace请选择CloudEC



----结束

eSDK CloudVC 20.1.RC1 开发指南(Android)

其他渠道

如果您在使用过程中有任何疑问，都可以通过以下方式联系我们。

1. 华为云开发者社区：**https://www.huaweicloud.com/?locale=zh-cn**
2. 华为云开发者社区（论坛）：**https://bbs.huaweicloud.com/forum/forum-943-1.html**

3.5 技术支持渠道

华为云开发者社区提单

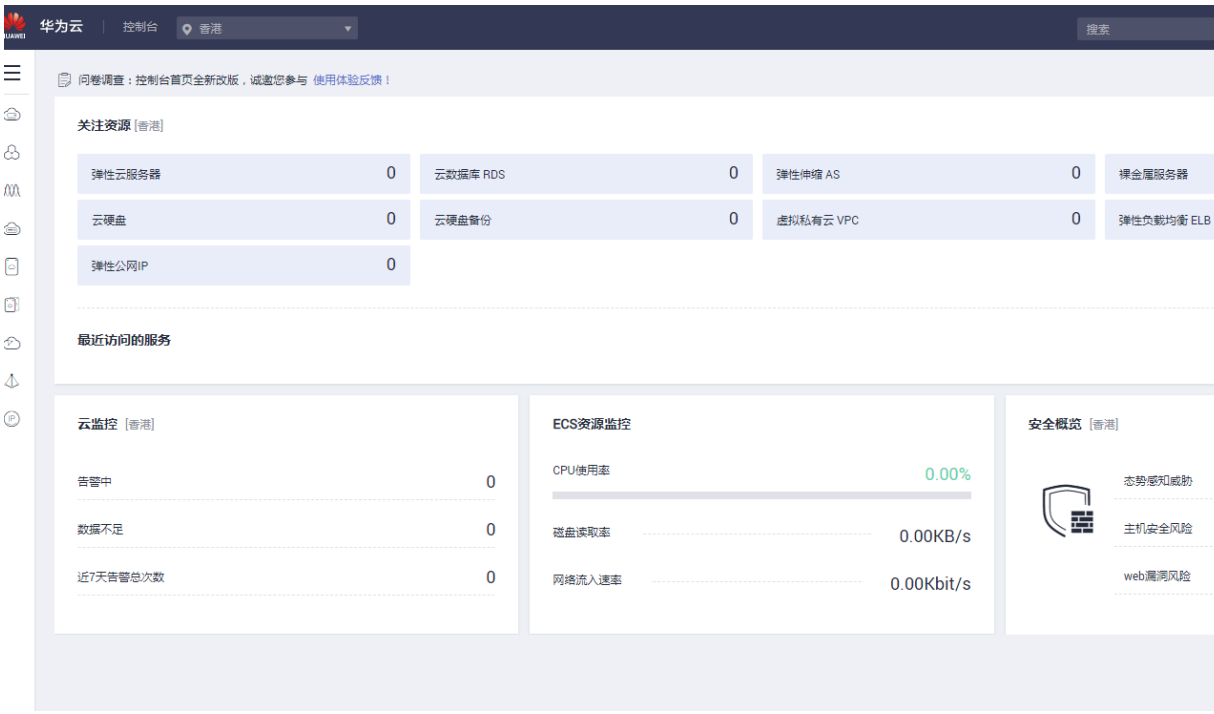
开发过程中，您有任何问题都可以在**华为云开发者社区**中提单跟踪。具体步骤如下：

步骤1 登录**华为云开发者社区**。

已注册有华为开发者社区帐号的，可直接登录。未注册的按照相关要求注册后登录。

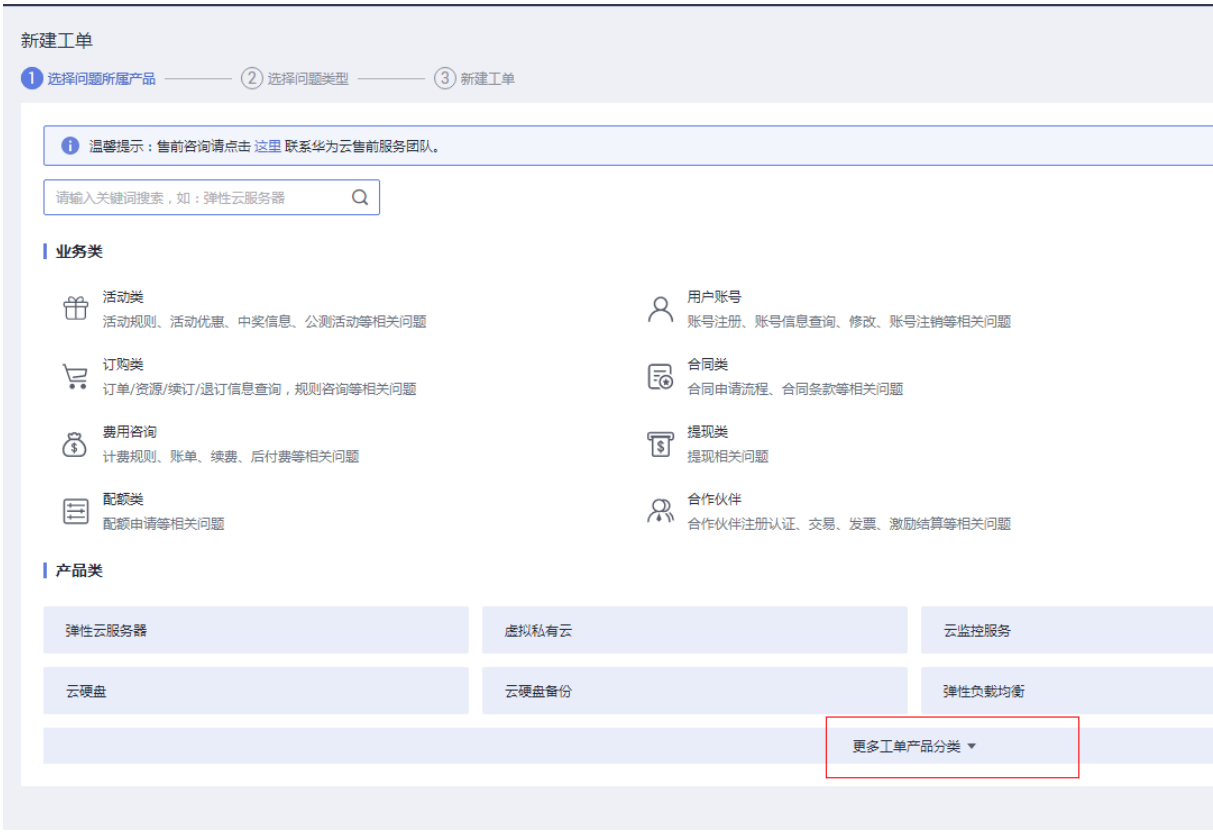
步骤2 单击“**控制台**”，进入控制台页面。

步骤3 在控制台中点击**工单，新建工单**



步骤4 在新建工单中，点击**更多工单产品分类。**

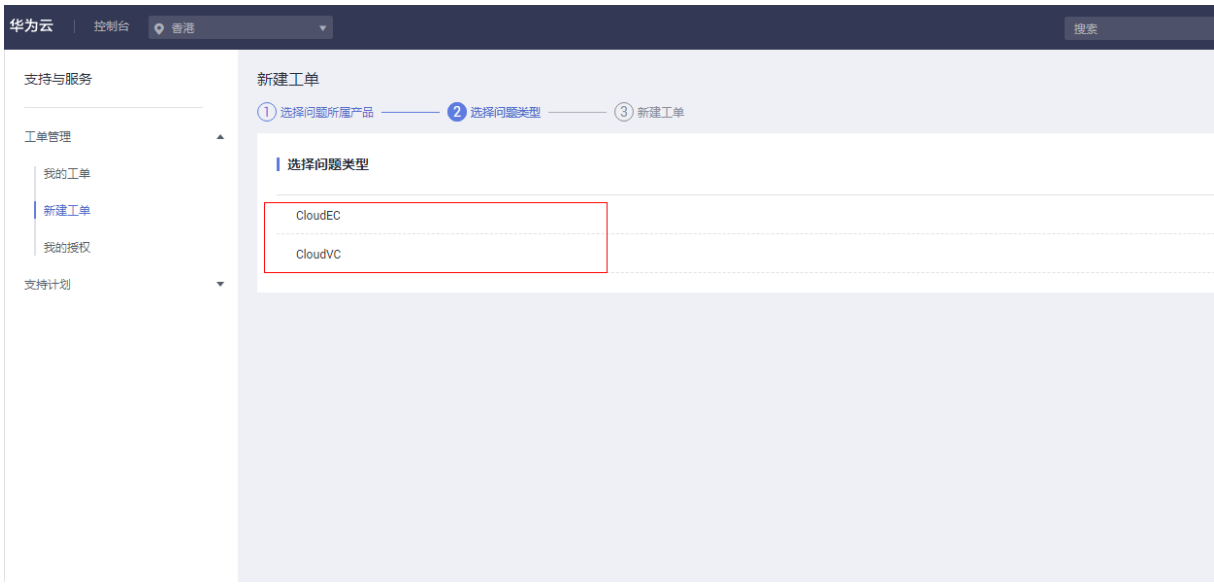
eSDK CloudVC 20.1.RC1 开发指南(Android)



在ICT开发者业务中选择**企业通信**

eSDK CloudVC 20.1.RC1 开发指南(Android)

根据产品选择**CloudEC或者CloudVC**进行提单，只有视频会议请选择CloudEC，如果有eSpace请选择CloudEC



----结束

eSDK CloudVC 20.1.RC1 开发指南(Android)

其他渠道

如果您在使用过程中有任何疑问，都可以通过以下方式联系我们。

1. 华为云开发者社区：**https://www.huaweicloud.com/?locale=zh-cn**
2. 华为云开发者社区（论坛）：**https://bbs.huaweicloud.com/forum/forum-943-1.html**

4 Hello World

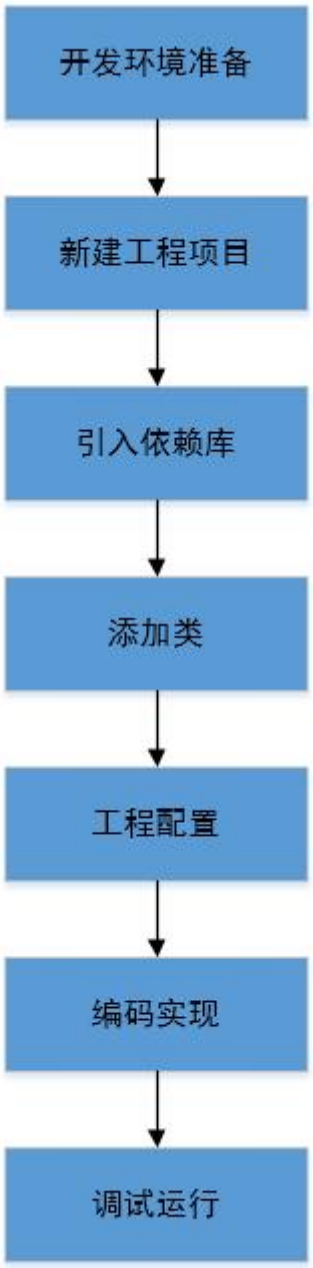
4.1 概述

Hello World 开发流程

本示例以java语言调用SDK接口，简单完成登录鉴权功能的开发，向您展示eSDK CloudVC Android的二次集成开发流程。

开发过程中的故障处理请参考章节**问题定位**。

快速入门流程如下：



4.2 环境准备

开发工具

- 操作系统：Windows 7专业版。
- Android Studio：Android Studio1.5及以上版本。

SDK 软件包和文档

SDK软件包名称：eSDK_EC_API_VX.X.X_Android.zip

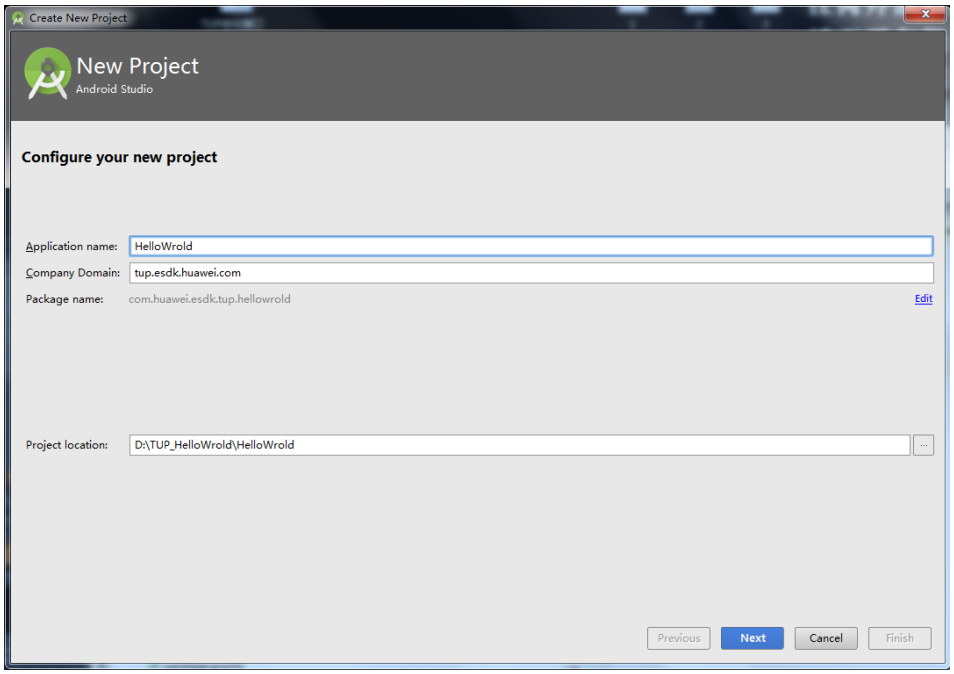
eSDK CloudVC 20.1.RC1 开发指南(Android)

文档名称：《CloudEC X.X.X 接口参考(Android)》、《CloudEC X.X.X 开发指南(Android)》

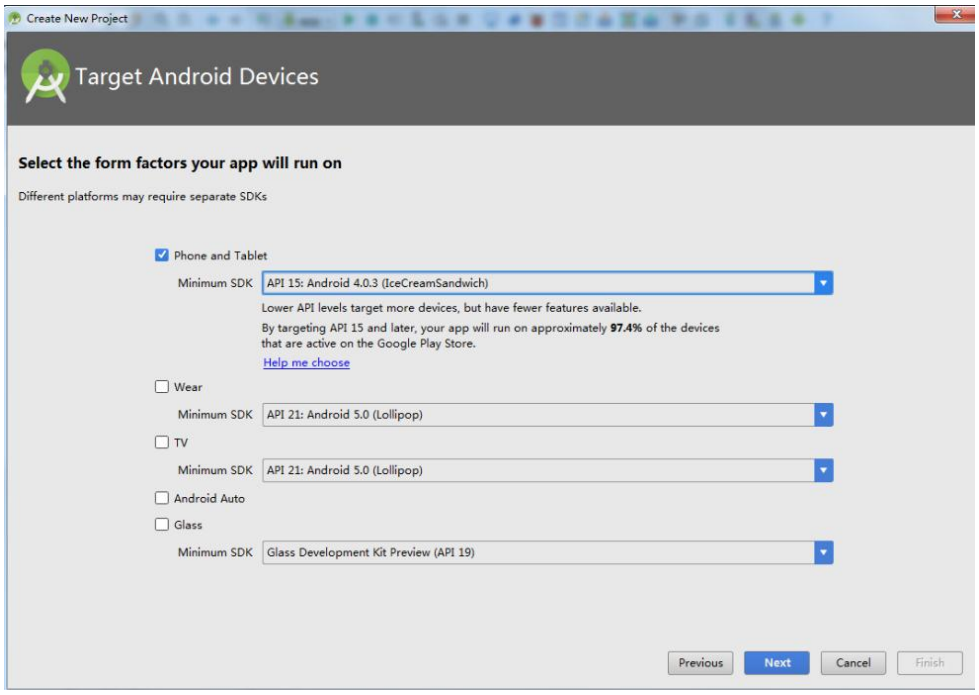
SDK和文档下载路径：参见**SDK和文档下载路径**。

4.3 新建工程项目

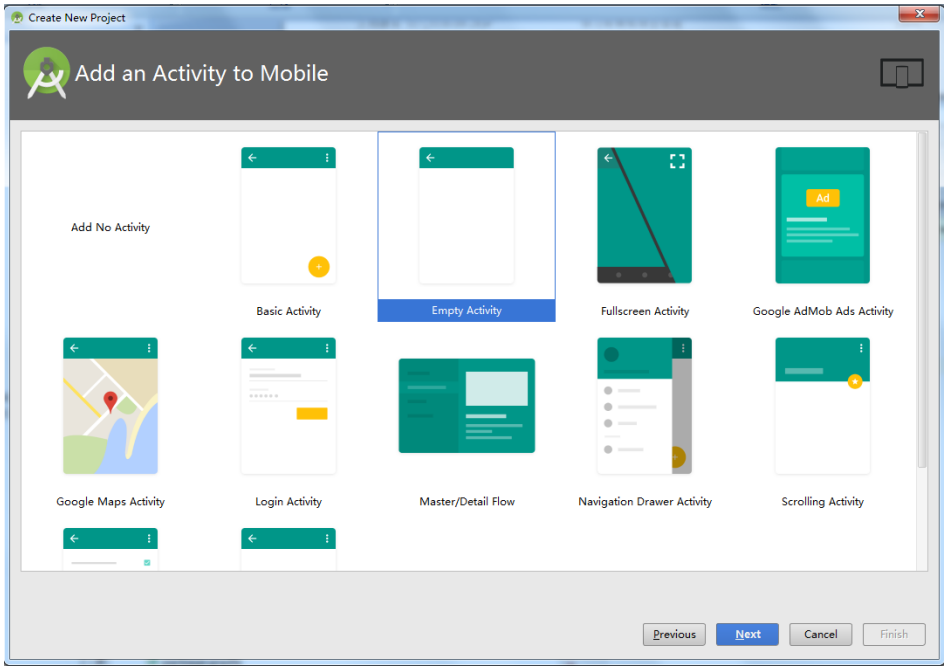
步骤1 打开Android Studio，选择“File > New > New Project… ”，系统显示“New Project”界面。



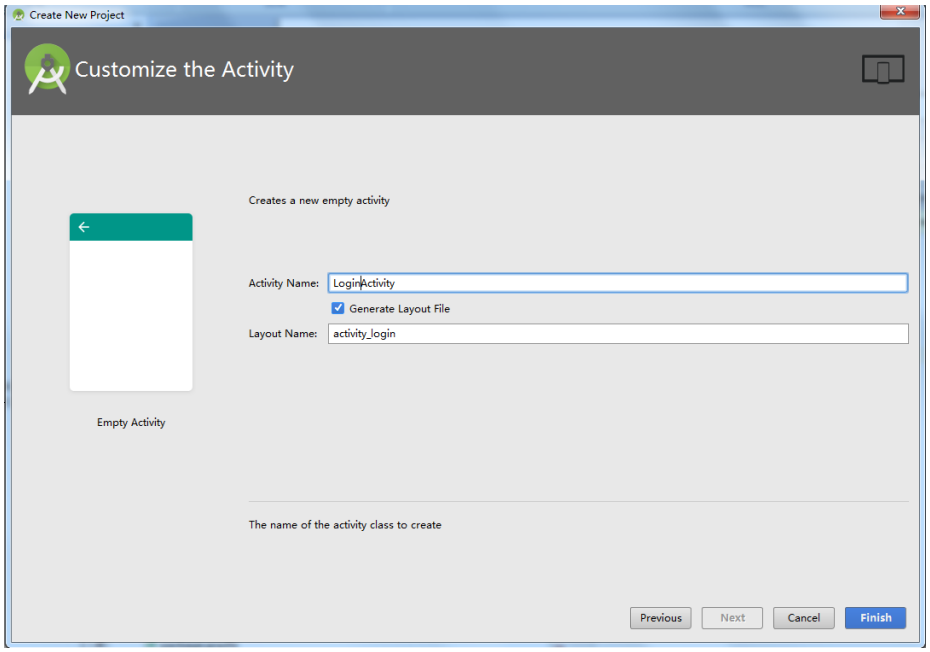
步骤2 填写Application name、Company Domain和Project location信息后，点击“Next”，进入SDK版本选择界面。



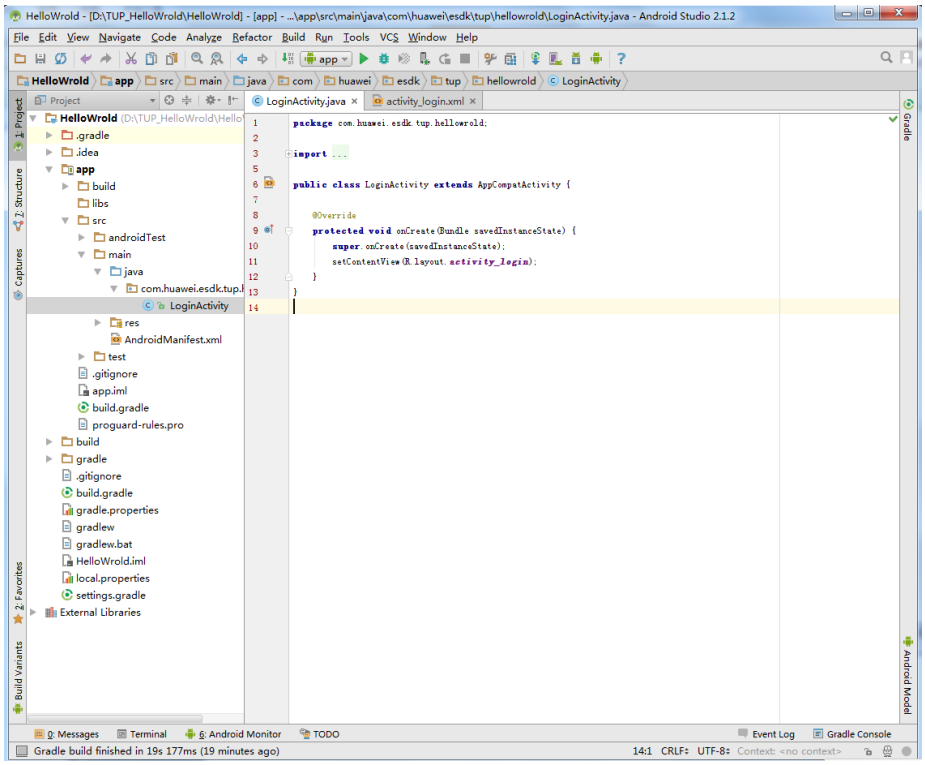
步骤3 选取最低支持系统的SDK版本后，点击“Next”，进入Activity展示效果界面。



步骤4 选取“Empty Activity”，点击“Next”，进入创建Activity界面。



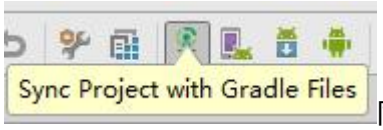
步骤5 在Activity Name和Layout Name分别填写“LoginActivity”和“activity_login”，点击“Finish”进入工程主界面。



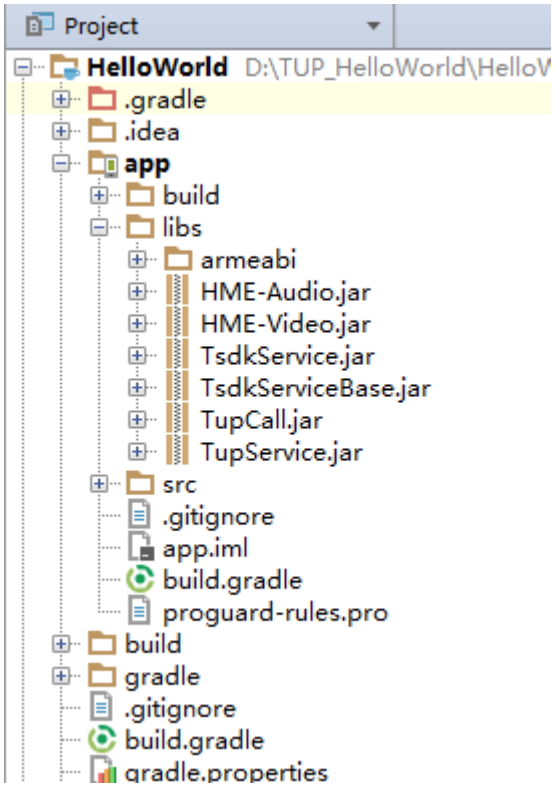
----结束

4.4 引入依赖库

将SDK压缩包解压后相关文件拷贝到工程的libs路径下。

so文件存放到“libs/armeabi”文件夹下，jar文件存放“libs”文件夹下，完成后单击“Sync” 

同步工程。

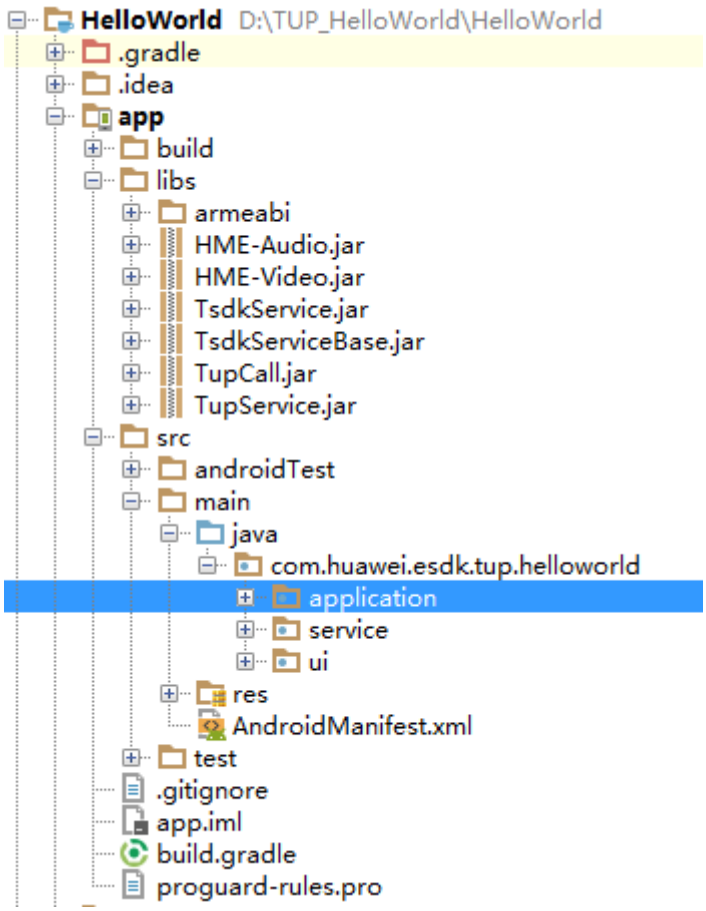


4.5 添加类

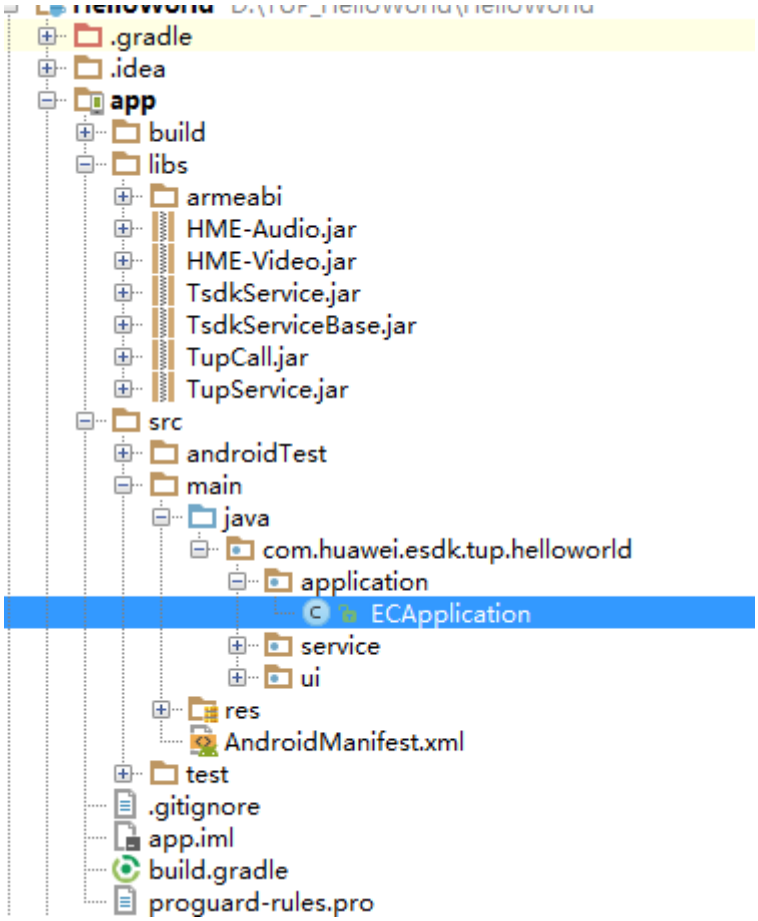
4.5.1 application

步骤1 右键单击已经新建的包名“com.huawei.esdk.tup.helloworld”文件夹，右键选择“New > Package”，弹出”New Package”窗口。在输入框中输入“application”，点击“OK”，完成包的新建。

eSDK CloudVC 20.1.RC1 开发指南(Android)



步骤2 右键“com.huawei.esdk.tup.helloworld.application”包，选择“New > Java Class”，弹出“Create New Class”窗口。在”Name”处输入“ECApplication”，点击“OK”，完成类的新建。

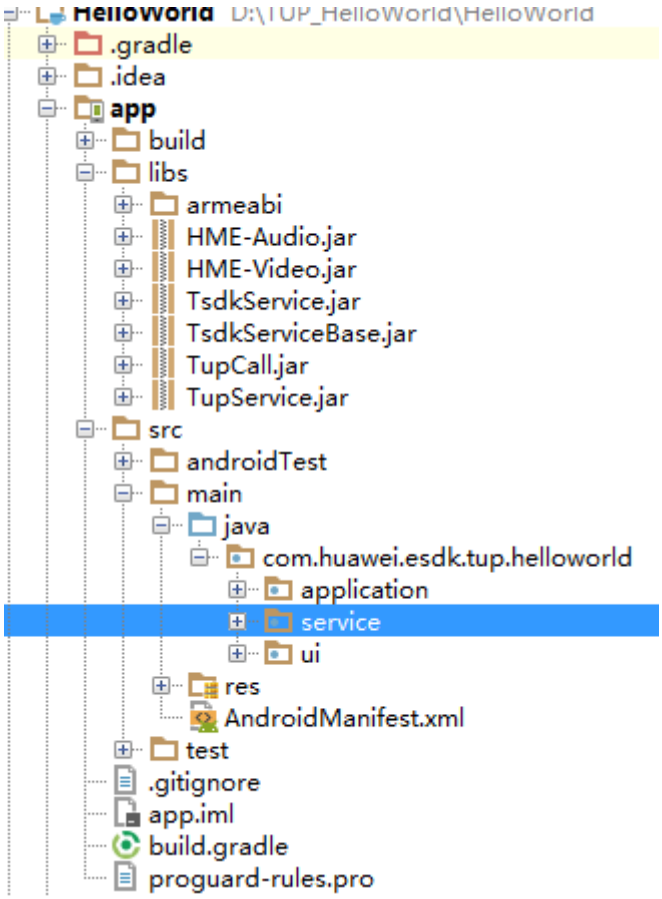


----结束

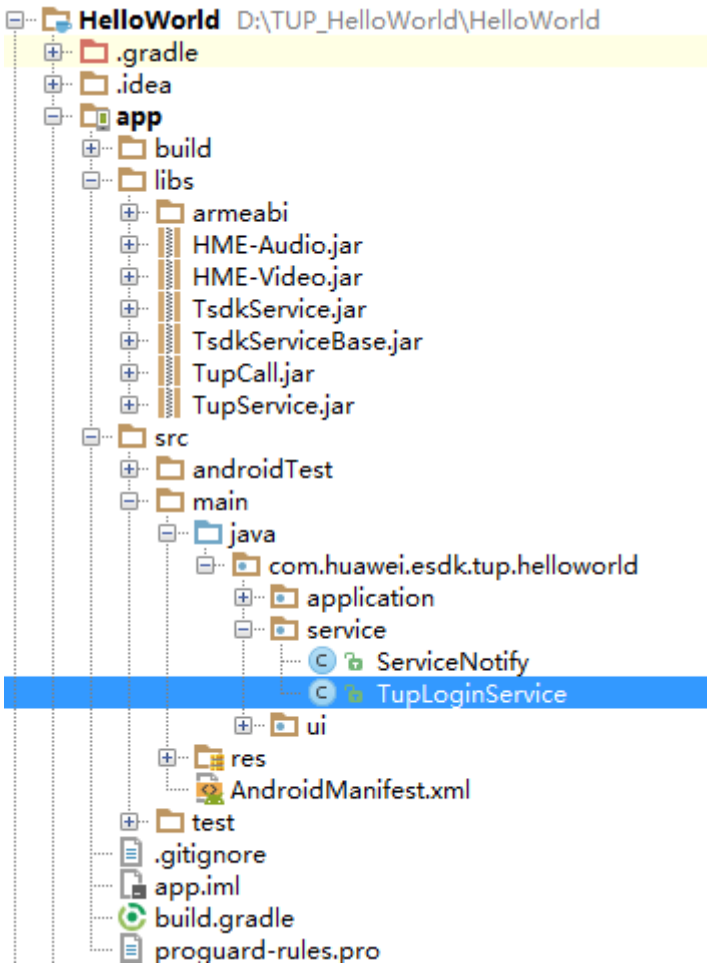
4.5.2 Service

步骤1 右键单击已经新建的包名“com.huawei.esdk.tup.helloworld”文件夹，右键选择“New > Package”，弹出”New Package”窗口。在输入框中输入“service”，点击“OK”，完成包的新建。

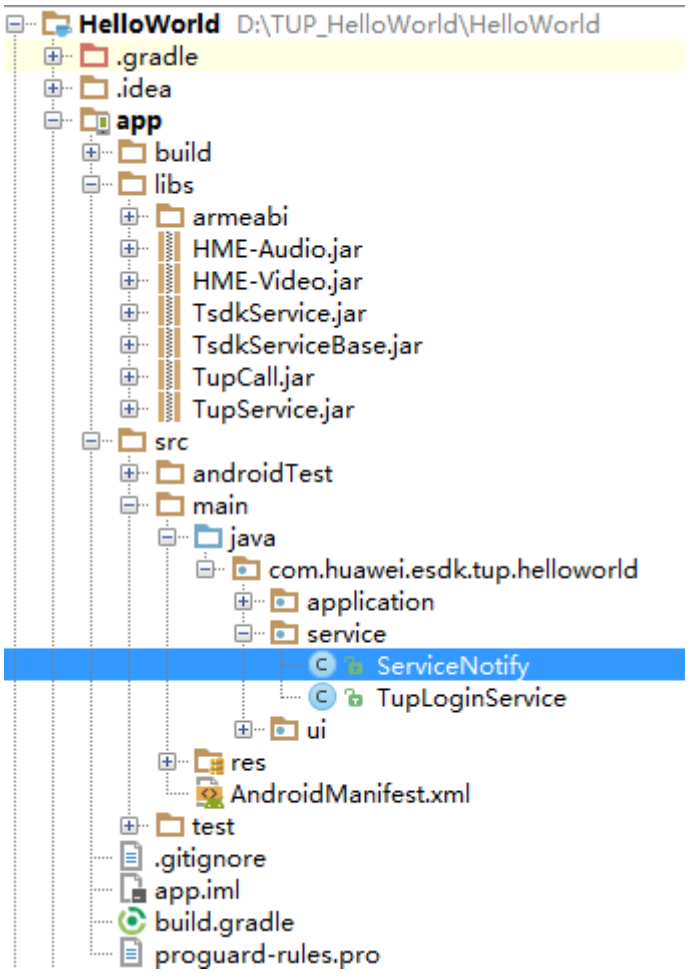
eSDK CloudVC 20.1.RC1 开发指南(Android)



步骤2 右键“com.huawei.esdk.tup.helloworld.service”包，选择“New > Java Class”，弹出“Create New Class”窗口。在”Name”处输入“TupLoginService”，点击“OK”，完成类的新建。



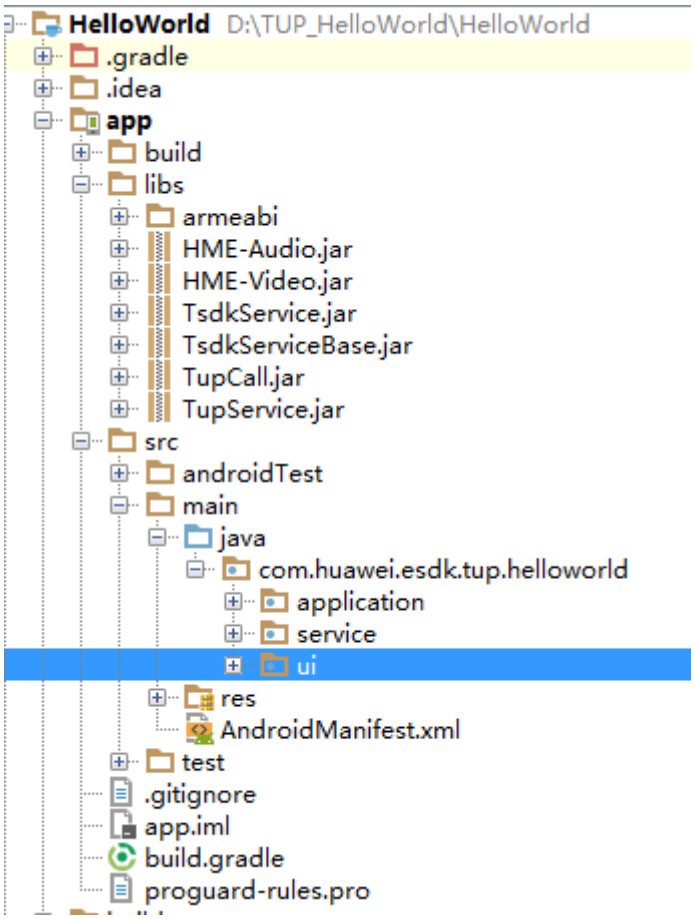
步骤3 右键“com.huawei.esdk.tup.helloworld.service”包，选择“New > Java Class”，弹出“Create New Class”窗口。在”Name”处输入“ServiceNotify”，点击“OK”，完成类的新建。



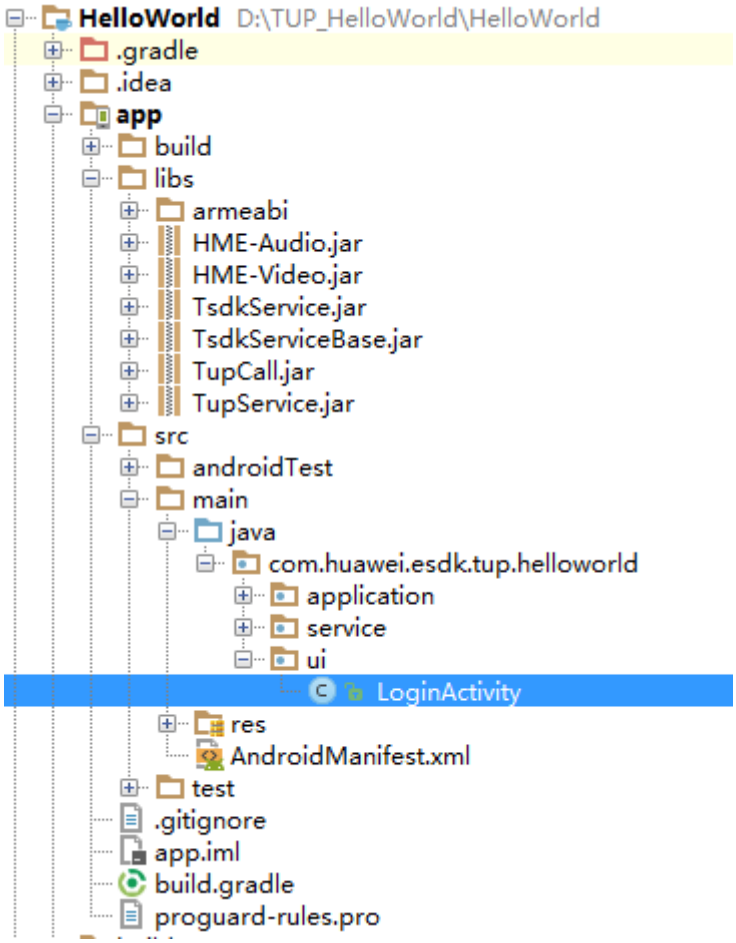
----结束

4.5.3 ui

步骤1 右键单击已经新建的包名“com.huawei.esdk.tup.helloworld”文件夹，右键选择“New > Package”，弹出”New Package”窗口。在输入框中输入“ui”，点击“OK”，完成包的新建。



步骤2 右键“com.huawei.esdk.tup.helloworld”包，选择“LoginActivity.java”右击复制，拷贝到"ui"目录下。



----结束

4.6 工程配置

步骤1 点击“res->values”，找到该文件夹下的“strings.xml”文件，添加配置代码。

```
<resources>
    <string name="app_name">HelloWrold</string>
    <string name="register_server">RegServer</string>
    <string name="server_port">ServerPort</string>
    <string name="account">Account</string>
    <string name="password">Password</string>
    <string name="login">Login</string>
    <string name="login_success">Login success</string>
    <string name="account_information_not_empty">Please enter your login parameters</string>
</resources>
```

步骤2 配置“app”文件夹下的“build.gradle”文件，工程创建后，在“build.gradle”中添加如下示例代码：

说明

targetSdkVersion版本需要在25及以下（targetSdkVersion 25）

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

android {
    compileSdkVersion 25
    buildToolsVersion "26.0.1"
```



```
    defaultConfig {
        jackOptions {
            enabled true
        }
    }

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

    defaultConfig {
        applicationId "com.huawei.esdk.tup.helloworld"
        minSdkVersion 15
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    sourceSets {
        main {
            assets.srcDirs = ['assets', 'src/main/assets', 'src/main/assets/']
            jniLibs.srcDirs = ['libs']
        }
        instrumentTest.setRoot('tests')
        debug.setRoot('build-types/debug')
        release.setRoot('build-types/release')
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    testCompile 'junit:junit:4.12'
    compile 'com.google.code.gson:gson:2.3.1'
}
```

步骤3 AndroidManifest.xml文件中添加如下权限代码：

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

----结束

4.7 编码实现

总体结构

```
--- src
 --- main
  --- java
    --- com.huawei.esdk.uc.helloworld.application
       --- ECApplication.java
    --- com.huawei.esdk.uc.helloworld.service
```

```
    ---TupLoginService.java
  --- com.huawei.esdk.uc.helloworld.ui
    ---LoginActivity.java
 ---res
  --- layout
   ---activity_login.xml
  --values
   ---strings.xml
---AndroidManifest.xml
```

源码链接：**Hello World源码文件**。

代码参考

- 在AndroidManifest.xml清单文件中指定ECApplication

```
<application
        android:name=".application.ECApplication"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true">
        <activity android:name=".ui.LoginActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

- 编写ECApplication类，初始化组件

```
    /**
     * Init int.
     * @param context the context
     * @param appPath the app path
     * @return the int
     */
    public int startService(Context context, String appPath)
    {
        tsdkManager = TsdkManager.getInstance(context, appPath, ServiceNotify.getInstance());
        TsdkAppInfoParam appInfoParam = new TsdkAppInfoParam();
        appInfoParam.setClientType(TsdkClientType.TSDK_E_CLIENT_MOBILE);
        appInfoParam.setProductName("SoftClient on Mobile");
        appInfoParam.setDeviceSn("123");
        appInfoParam.setSupportAudioAndVideoCall(0);
        appInfoParam.setSupportAudioAndVideoConf(0);
        appInfoParam.setSupportDataConf(0);
        appInfoParam.setSupportCtd(0);
        appInfoParam.setSupportEnterpriseAddressBook(0);
        appInfoParam.setSupportIm(0);
        appInfoParam.setSupportRichMediaMessage(0);

        int ret = tsdkManager.init(appInfoParam);
        return ret;
    }
```

- 去初始化

```
    @Override
    public void onTerminate() {
        super.onTerminate();
        tsdkManager.uninit();
    }
```

- 编写TupLoginService服务类，Uportal鉴权登录

```
    /**
     * Authorize login boolean.
     *
     * @param regServer  the reg server
     * @param serverPort the server port
```



```
    * @param sipNumber  the sip number
    * @param password   the password
    * @return the boolean
    */
   public boolean authorizeLogin(String regServer, String serverPort, String sipNumber, String
password)
   {
      int ret;

      TsdkLoginParam tsdkLoginParam = new TsdkLoginParam();
      tsdkLoginParam.setUserId(1);
      tsdkLoginParam.setAuthType(TsdkAuthType.TSDK_E_AUTH_NORMAL);
      tsdkLoginParam.setUserName(sipNumber);
      tsdkLoginParam.setPassword(password);
      tsdkLoginParam.setServerAddr(regServer);
      tsdkLoginParam.setServerPort(Integer.parseInt(serverPort));
      tsdkLoginParam.setServerVersion("");
      tsdkLoginParam.setServerType(TsdkServerType.TSDK_E_SERVER_TYPE_PORTAL);
      tsdkLoginParam.setUserTiket("");

      ret = TsdkManager.getInstance().getLoginManager().login(tsdkLoginParam);
      if (ret != 0) {
         return false;
      }
      return true;
   }
```

- ServiceNotify实现TsdkNotify后重写onEvtAuthSuccess()方法，鉴权成功发送广播。

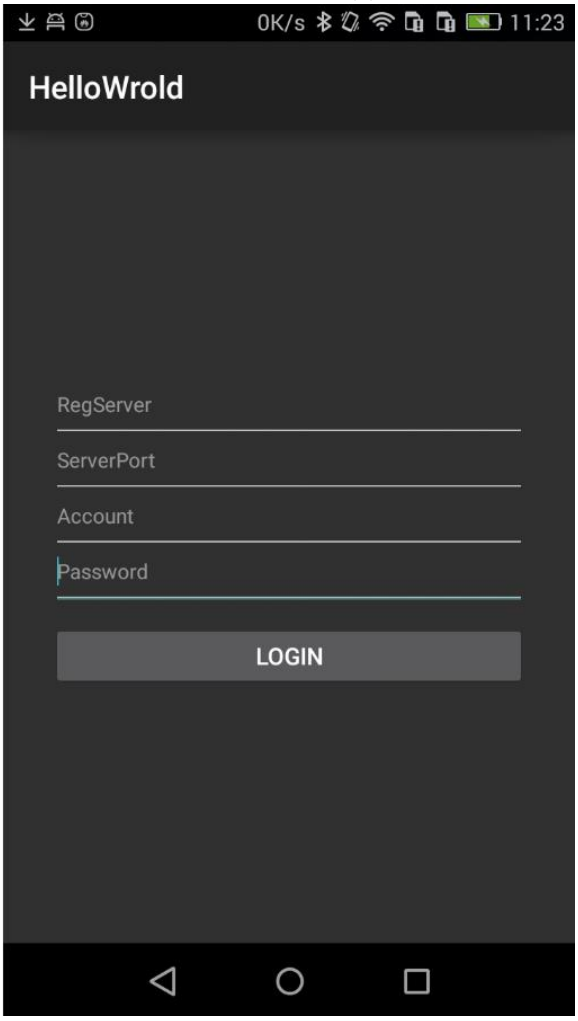
```
   @Override
   public void onEvtAuthSuccess(int i, TsdkImLoginParam tsdkImLoginParam) {
      TupLoginService.getInstance().handleAuthSuccess();
   }
   public void handleAuthSuccess() {
      Log.d("TupLoginService", "onEvtAuthSuccess.");

      Intent intent = new Intent();
      intent.setAction(TUP_REGISTER_EVENT);
      intent.putExtra(REGISTER_RESULT, 0);
      ECApplication.getApp().sendBroadcast(intent);
   }
```

- UI层注册监听广播，保证调用鉴权接口后收到相应回调。

```
   private BroadcastReceiver receiver = new BroadcastReceiver() {
      @Override
      public void onReceive(Context context, Intent intent) {
         String action = intent.getAction();
         if (TupLoginService.TUP_REGISTER_EVENT.equals(action))
         {
            int result = intent.getIntExtra(TupLoginService.REGISTER_RESULT, -1);
            switch (result)
            {
               case 0:
                  Toast.makeText(LoginActivity.this, getString(R.string.login_success),
Toast.LENGTH_LONG).show();
                  break;
               default:
                  break;
            }
         }
      }
   };
```

4.8 调试运行

步骤1 点击菜单栏“Run > Run > app”文件，选择真机调试或者运行虚拟机。



步骤2 在界面中依次输入如下表信息：

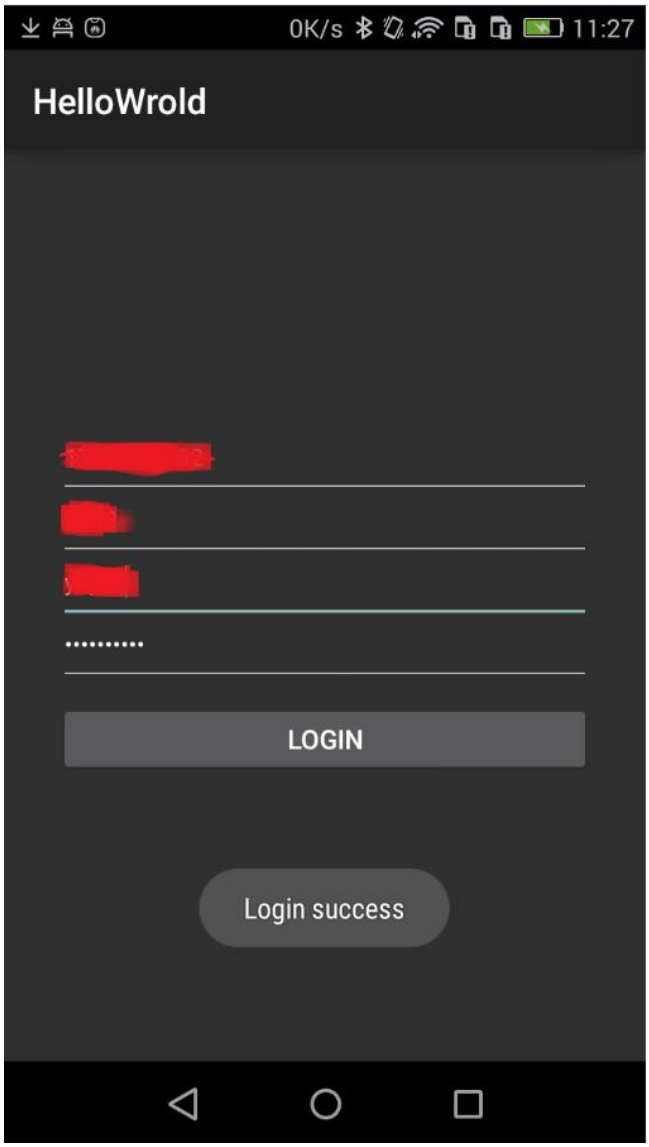
分类	输入信息	说明
鉴权信息	RegServer	注册服务器地址
	Server port	服务器端口号
	Account	服务号码
	Password	eSDK鉴权密码

说明

以上信息需要在成功预约华为远程实验室后，从远程实验室获取。

服务器地址、端口号、用户名、密码输入完成后，单击“登录”按钮，界面显示“正在登录，请稍后”，等待接口返回操作结果。

步骤3 等待至页面底部弹出Toast“Login Success”，表示登录成功。



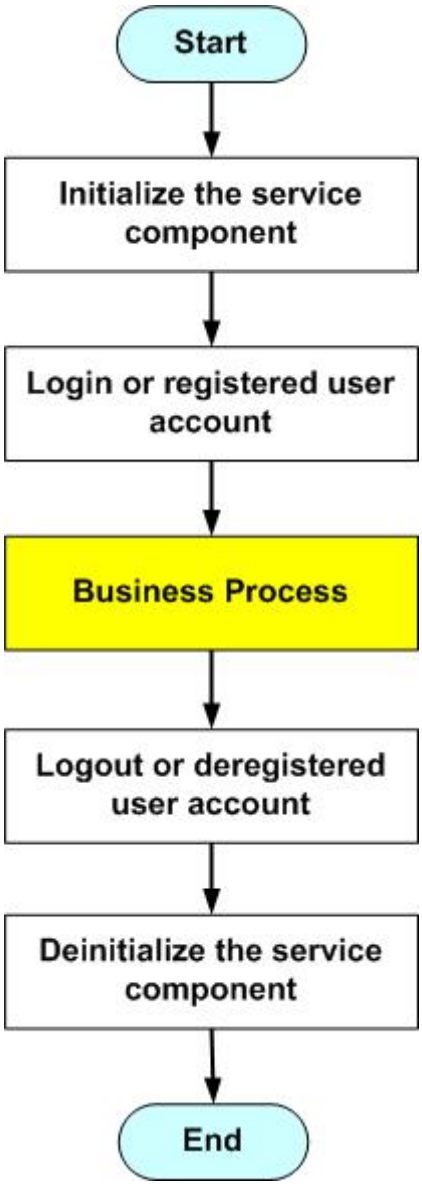
----结束

eSDK CloudVC 20.1.RC1 开发指南(Android)

5 业务开发指导

5.1 业务开发总体流程

您在使用eSDK CloudVC提供的各类组件业务接口集成开发时，基本的流程步骤包含：初始化业务组件、登录或注册用户帐号、业务实现调用、登出或注销用户帐号和去初始化业务组件。



流程说明

1. **初始化组件**：实现对业务组件进行资源初始化，设置第三方应用程序的全局业务配置参数。
2. **登录或注册用户帐号**：先完成本地地址、端口和服务器地址、端口等登录或注册相关参数设置，再调用相应接口完成向服务器的注册。
3. **业务实现调用**：根据实际业务开发需要，调用相应的业务接口实现相关业务，具体可参考后继典型业务场景描述的相关章节。
4. **登出或注销用户帐号**：实现用户退出登录或注册，确保业务接口使用的安全性。
5. **去初始化组件**：实现对业务组件占用的资源释放。

5.2 组件初始化

应用场景

在使用eSDK CloudVC系列业务，配套CloudVC解决方案实现各类业务前，需要先完成SDK初始化。

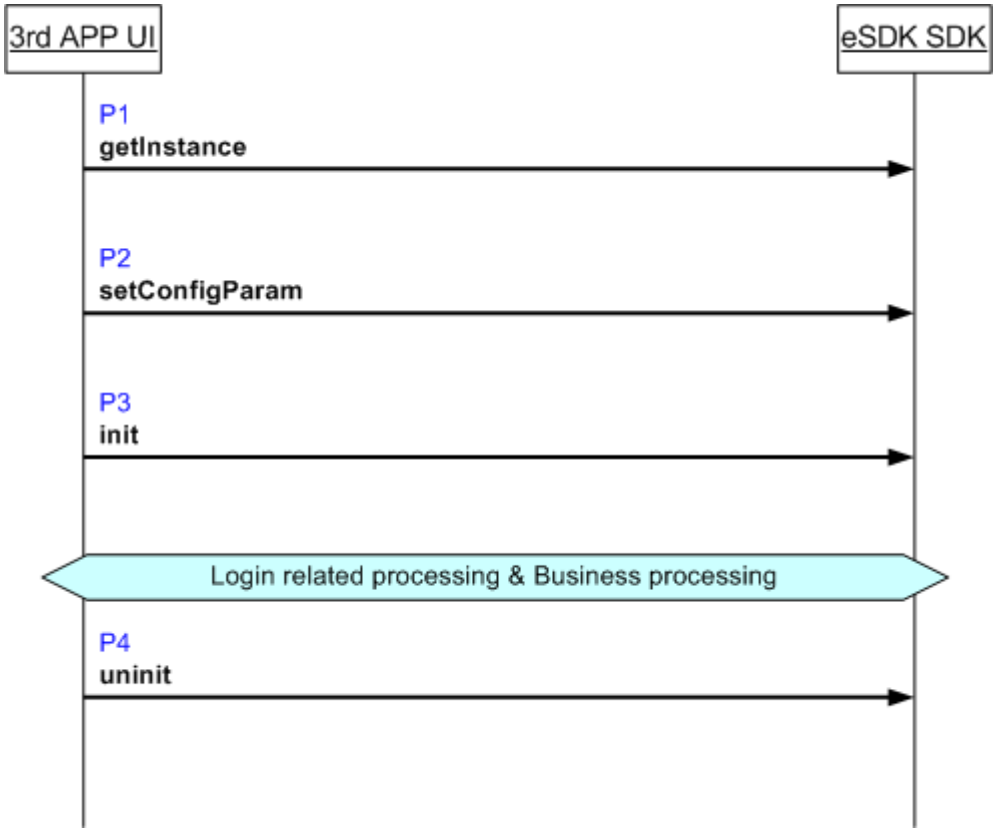
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

无。

流程说明

图 5-1 初始化和去初始化组件流程图



初始化

步骤1 应用程序在初始化组件前，调用TsdkManager类的getInstance()方法得到实例对象并且初始化对象的值。

说明

在得到实例对象的同时需要传入一些必要的参数，上下文(context),应用程序库加载路径(appPath),SDK 事件通知处理对象(notify).

代码示例：

```
//java code
TsdkManager tsdkManager = TsdkManager.getInstance(context, appPath, ServiceNotify.getInstance());
```

步骤2 应用程序在初始化组件前，调用TsdkManager类的setConfigParam()方法设置业务参数。

说明

除特定的必选参数外，应用程序若不进行相应参数设置，组件则使用默认配置。在初始化前设置的参数包括：

1. 日志参数，对应的配置ID：TSDK_E_CONFIG_LOG_PARAM，对应的参数：TsdkLogParam，此参数在iOS平台必选；
2. TLS参数，对应的配置ID：TSDK_E_CONFIG_TLS_PARAM，对应的参数：TsdkTlsParam；
3. 应用程序文件路径信息，对应的配置ID：TSDK_E_CONFIG_APP_FILE_PATH_INFO，对应的参数：TsdkAppFilePathInfo，此参数在使用“企业通讯录”功能时必选；
4. 设备DPI信息，对应的配置ID：TSDK_E_CONFIG_DEVICE_DPI_INFO，对应的参数：TsdkDeviceDpiInfo。
5. 设置会议控制参数，对应的配置ID：TSDK_E_CONFIG_CONF_CTRL_PARAM，对应的数据结构：TsdkConfCtrlParam。

代码示例：

```
//java code
TsdkLogParam logParam = new TsdkLogParam();
logParam.setFileCount(1);
logParam.setLevel(TsdkLogLevel.TSDK_E_LOG_DEBUG);
logParam.setMaxSizeKb(1024 * 4);
logParam.setPath(Environment.getExternalStorageDirectory() + File.separator + "ECSDKDemo" + "/");

int ret = TsdkManager.getInstance().setConfigParam(logParam);
```

步骤3 应用程序调用TsdkManager类的init()方法实现组件初始化。

说明

1. 应用程序信息参数(TsdkAppInfoParam)，包含客户端类型、产品信息以及当前应用程序支持的功能，SDK将根据相应的信息完成初始化：
   1. 对于PC客户端，终端类型(client_type)应取值TSDK_E_CLIENT_PC；对于移动客户端，终端类型(client_type)应取值TSDK_E_CLIENT_MOBILE；
   2. 产品名信息，标识应用程序的类型，取值如“eSDK-Mobile”，对于可能存在的EC服务器特定的配置，此值存在差异，若填写与服务器配置不匹配，会导致登录过程失败。
2. 应用程序关注的事件：

事件ID	事件说明
onEvtAuthSuccess	鉴权成功(用于呈现登录过程，应用层一般无需处理)
onEvtAuthFailed	鉴权失败。
onEvtAuthRefreshFailed	鉴权刷新失败。

事件ID	事件说明
onEvtLoginSuccess	登录成功。
onEvtLoginFailed	登录失败。
onEvtLogoutSuccess	登出成功。
onEvtLogoutFailed	登出失败。
onEvtForceLogout	强制登出。
onEvtVoipAccountStatus	VOIP帐号信息。
onEvtFirewallDetectFailed	防火墙探测失败。
onEvtBuildStgTunnelFailed	创建stg通道失败。
onEvtSecurityTunnelInfoInd	安全隧道信息通知。
onEvtGetTempUserResult	获取用于匿名方式加入会议的临时用户结果通知。
onEvtCallStartResult	发起呼叫结果。
onEvtCallIncoming	来电事件。
onEvtCallOutgoing	呼出事件。
onEvtCallRingback	回铃音事件(在需要APP播放回铃音时上报)。
onEvtCallRtpCreated	RTP通道已建立，可以进行二次拨号。
onEvtCallConnected	通话已建立。
onEvtCallEnded	呼叫结束。
onEvtCallDestroy	呼叫结束后销毁呼叫控制信息。
onEvtOpenVideoReq	远端请求打开视频。
onEvtRefuseOpenVideoInd	远端拒绝请求打开视频通知。
onEvtCloseVideoInd	关闭视频(视频转音频)通知。
onEvtOpenVideoInd	打开视频(音频转视频)通知。
onEvtRefreshViewInd	视频view刷新通知。
onEvtCallRouteChange	移动路由变化通知。
onEvtPlayMediaEnd	音频文件播放结束通知。
onEvtSessionModified	会话修改完成通知。
onEvtSessionCodec	会话正在使用的codec通知。
onEvtHoldSuccess	呼叫保持成功。
onEvtHoldFailed	呼叫保持失败。

事件ID	事件说明
onEvtUnholdSuccess	恢复通话成功。
onEvtUnholdFailed	恢复通话失败。
onEvtEndcallFailed	结束通话失败。
onEvtDivertFailed	偏转失败。
onEvtBldTransferSuccess	盲转成功。
onEvtBldTransferFailed	盲转失败。
onEvtSetIptServiceResult	登记IPT业务结果。
onEvtIptServiceInfo	ipt业务信息变更通知。
onEvtBookConfResult	预约会议结果。
onEvtQueryConfListResult	查询会议列表结果。
onEvtQueryConfDetailResult	查询会议详情结果。
onEvtJoinConfResult	加入会议结果。
onEvtGetDataconfParamResult	获取数据会议参数结果。
onEvtConfctrlOperationResult	会控操作结果。
onEvtInfoAndStatusUpdate	会议信息及状态更新。
onEvtSpeakerInd	发言方通知。
onEvtRequestConfRightFailed	申请会控权限失败。
onEvtConfIncomingInd	会议来电通知。
onEvtConfEndInd	会议结束通知。
onEvtJoinDataConfResult	加入数据会议结果。
onEvtAsStateChange	应用共享状态通知
onEvtRecvChatMsg	收到会议中的聊天消息通知
onEvtTransToConfResult	呼叫转成会议结果
onEvtDsDocNew	新建一个共享文档。
onEvtDsDocDel	删除一个共享文档。
onEvtWbDocNew	新建一个白板文档。
onEvtWbDocDel	删除一个白板文档。
onEvtCtdStartCallResult	发起ctd呼叫结果。
onEvtCtdEndCallResult	结束ctd呼叫结果。
onEvtCtdCallStatusNotify	ctd呼叫状态上报。

事件ID	事件说明
onEvtSearchContactsResult	查询联系人结果。
onEvtSearchDeptResult	查询部门结果。
onEvtGetIconResult	获取头像结果。

代码示例：

```
//Java code
TsdkAppInfoParam appInfoParam = new TsdkAppInfoParam();
appInfoParam.setClientType(TSDK_E_CLIENT_MOBILE);
appInfoParam.setProductName("SoftClient on Mobile");
appInfoParam.setDeviceSn("123");
appInfoParam.setSupportAudioAndVideoCall(this.isSupportAudioAndVideoCall?1:0);
appInfoParam.setSupportAudioAndVideoConf(this.isSupportAudioAndVideoConf?1:0);
appInfoParam.setSupportDataConf(this.isSupportDataConf?1:0);
appInfoParam.setSupportCtd(this.isSupportCTD?1:0);
appInfoParam.setSupportEnterpriseAddressBook(this.isSupportAddressbook?1:0);
appInfoParam.setSupportIm(0);
appInfoParam.setSupportRichMediaMessage(0);

int ret = TsdkManager.getInstance().init(appInfoParam);
```

----结束

去初始化

步骤1 应用程序关闭时，UI调用TsdkManager类的uninit()方法去初始化基础组件，释放相应资源。

代码示例：

```
//Java code
public int uninit();
```

----结束

注意事项

无。

5.3 登录与注销

应用场景

在使用CloudVC解决方案下的各类业务之前，需要向服务器完成登录；在不再使用业务时注销，确保业务接口使用的安全性。

说明

登录成功后，SDK自动按保活周期定时刷新鉴权凭证信息。

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

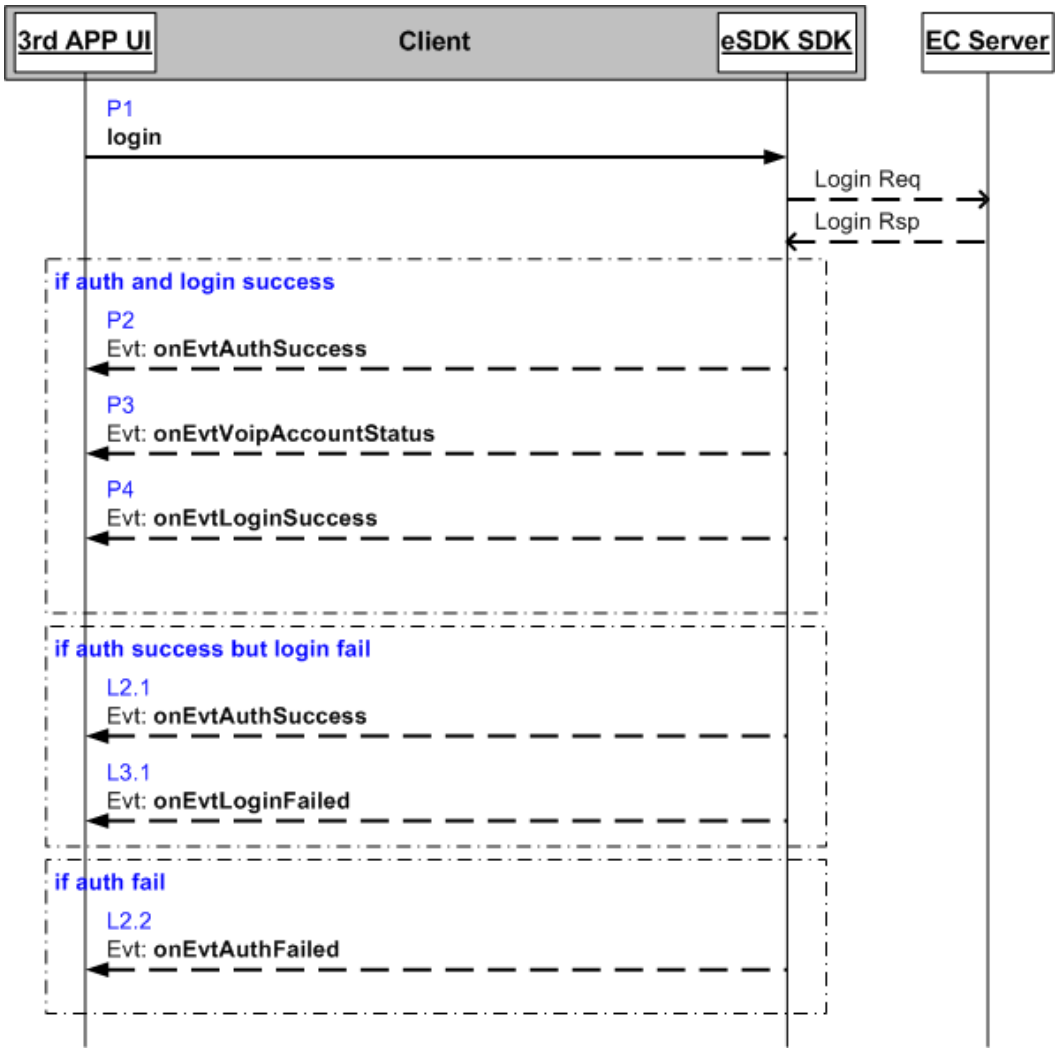
前提条件

已完成初始化。

流程说明

用户登录

图 5-2 登录处理流程



步骤1 UI调用TsdkLoginManager的login()方法进行登录，参数为TsdkLoginParam类。包括用户ID、鉴权类型、用户帐户和密码(或Tiket)以及服务器信息。

说明

1. 用户ID，由应用程序生成的标识，用于关联用户帐户；
2. 如果用户选择使用“密码鉴权”登录，则鉴权类型取值为TSDK_E_AUTH_TYPE，帐户的用户名和密码必需填写；如果用户选择使用第三方认证登录，则鉴权类型取值为TSDK_E_AUTH_TICKET，帐户的ticket值必须填写，取值为第三方提供的token值3rd_Token；
3. 服务器类型取值TSDK_E_SERVER_TYPE，暂仅支持TSDK_E_SERVER_TYEP_SMC。

代码示例：

```
//Java code
TsdkLoginParam tsdkLoginParam = new TsdkLoginParam();
tsdkLoginParam.setUserId(1);
tsdkLoginParam.setAuthType(TsdkAuthType.TSDK_E_AUTH_NORMAL);
tsdkLoginParam.setUserName(loginParam.getUserName());
tsdkLoginParam.setPassword(loginParam.getPassword());
tsdkLoginParam.setServerAddr(loginParam.getServerUrl());
tsdkLoginParam.setServerPort(loginParam.getServerPort());
tsdkLoginParam.setServerVersion("");
tsdkLoginParam.setServerType(TSDK_E_SERVER_TYEP_SMC);
tsdkLoginParam.setUserTicket("");

ret = TsdkManager.getInstance().getLoginManager().login(tsdkLoginParam);
```

步骤2 SDK收到服务器的鉴权登录响应后，向UI上报鉴权成功事件onEvtAuthSuccess。

说明

1. 如果鉴权失败，将不能进行下一步操作，也不会有业务账号和配置信息上报。

代码示例：

```
//Java code
public void onEvtAuthSuccess(int userId, TsdkImLoginParam imLoginParam) {
    LogUtil.e(TAG, "authorize success.");

}
```

步骤3 SDK收到服务器的鉴权登录响应后，向UI上报VOIP帐号信息事件onEvtVoipAccountStatus。

说明

1. 如果登录成功，会上报账号短号号码，UI应保存此号码，以方便后续操作。

代码示例：

```
//Java code
public void onEvtVoipAccountStatus(int userId, TsdkVoipAccountInfo voipAccountInfo ) {
    LogUtil.e(TAG, "voip account status: " );

    this.sipNumber = voipAccountInfo.getNumber();
    if (!voipAccountInfo.getTerminal().equals("")) {
        this.terminal = voipAccountInfo.getTerminal();
    }
}
```

步骤4 登录成功之后，SDK向UI上报登录成功事件onEvtLoginSuccess，UI做相应的界面处理。

说明

有时候会出现鉴权成功，但是voip登录失败的情况。可修改初始化时的参数product_name来解决。

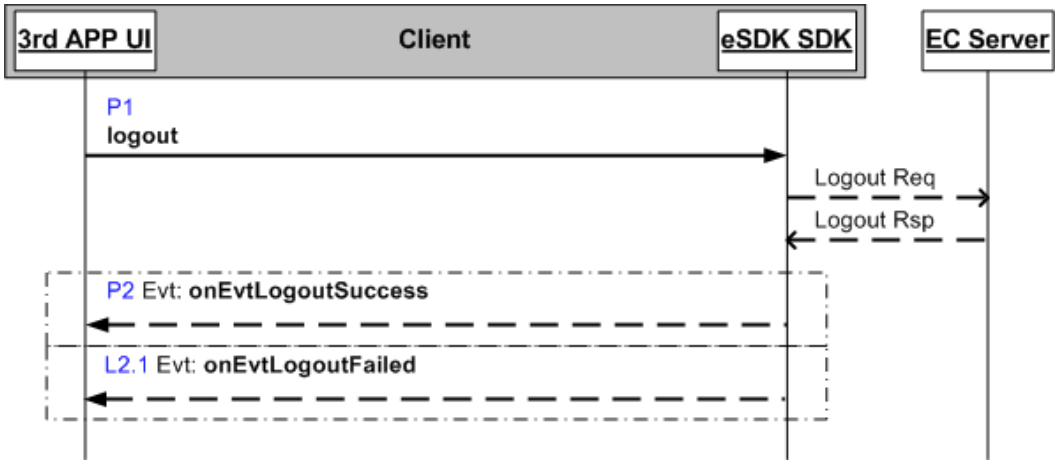
代码示例：

```
//Java code
public void onEvtLoginSuccess(int userId) {
    LogUtil.i(TAG, "voip login success");
    this.loginEventNotifyUI.onLoginEventNotify(LoginConstant.LoginUIEvent.LOGIN_SUCCESS, userId, "voip
login success");
}
```

----结束

用户主动注销

图 5-3 注销处理流程



步骤1 UI调用TsdkLoginManager的logout()方法发起注销。

代码示例：

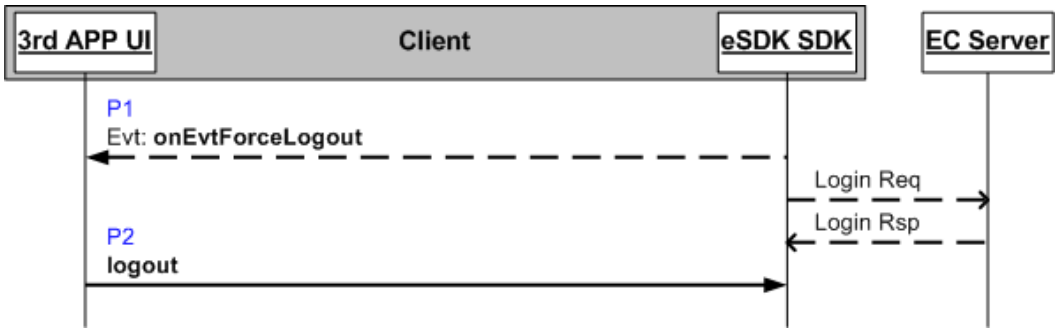
```
//Java code
public void logout() {
    int ret = TsdkManager.getInstance().getLoginManager().logout();
    if (ret != 0) {
        LogUtil.e(TAG, "login is failed, return " + ret);
    }

}
```

步骤2 登出成功之后，向UI上报登出成功事件onEvtLogoutSuccess。

----结束

服务器强制注销

图 5-4 服务器强制注销处理流程



说明

用户帐号在其他位置登录时，服务器会通知应用程序注销本地帐号。

步骤1 SDK收到服务器的强制登出通知消息后，向UI上报强制登出事件onEvtForceLogout。

步骤2 UI调用TsdkLoginManager的logout()方法完成登出过程。

----结束

断网重连

说明

应用程序监测到断网重连，应根据预先配置的策略确定是否自动发起登录流程，若预配置，则发起登录流程，与普通的“登录”流程相同。

注意事项

无。

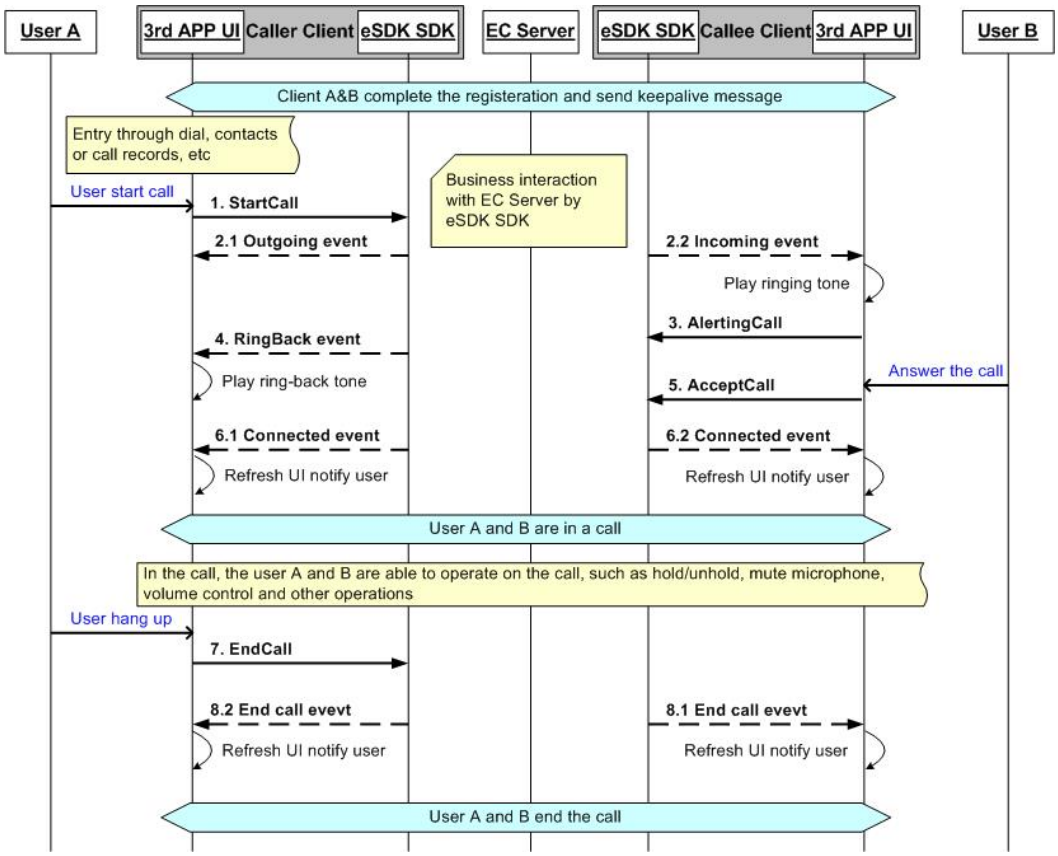
5.4 音视频呼叫

5.4.1 概述

SDK基于CloudVC解决方案，向第三方开发者开放音视频呼叫能力。

基本业务流程

音视频通话基本业务流程：



前提条件

主被叫应用程序完成向EC Server登录，并按周期自动发送登录保活消息。

流程说明

建立通话

步骤1 用户A通过拨号盘、联系人或通话记录等入口拨号呼出，主叫应用程序UI 调用SDK接口发起呼叫请求。

步骤2 主叫侧SDK向EC Server发送呼叫请求消息成功后，向UI上报呼出事件；被叫侧SDK收到EC Server呼叫请求消息后，向UI上报来电事件。

步骤3 被叫侧UI收到来电事件，播放来电提示音呈现来电通知界面，同时调用SDK接口发送消息通知主叫，本地正在振铃。

步骤4 主叫侧SDK收到被叫振铃通知消息，向UI上报回铃音事件，UI播放回铃音。

步骤5 被叫用户B选择接听来电，被叫侧UI调用SDK接口发送接听呼叫响应消息接听呼叫。

步骤6 主叫侧SDK收到被叫侧的呼叫响应消息后，与被叫侧SDK完成音视频通话协商，建立音视频通话，主、被叫侧SDK向UI上报呼叫接通事件，UI刷新界面通知用户通话接通。

说明

通话过程中，主叫用户A和B都可以对通话进行操作，如保持\恢复通话、静音麦克风和调整通话音量等。

----结束

挂断通话

说明

通话过程中，主叫或被叫用户均可以挂断通话，当前以主叫用户A发起挂断为例。

步骤1 用户A在通话界面挂断通话，应用程序UI调用SDK接口挂断呼叫，SDK发送挂断呼叫消息。

步骤2 主、被叫侧SDK完成结束呼叫处理，向UI上报呼叫结束事件，UI刷新界面通知用户通话结束。

----结束

5.4.2 建立音频通话

应用场景

用户点对点音频通话。

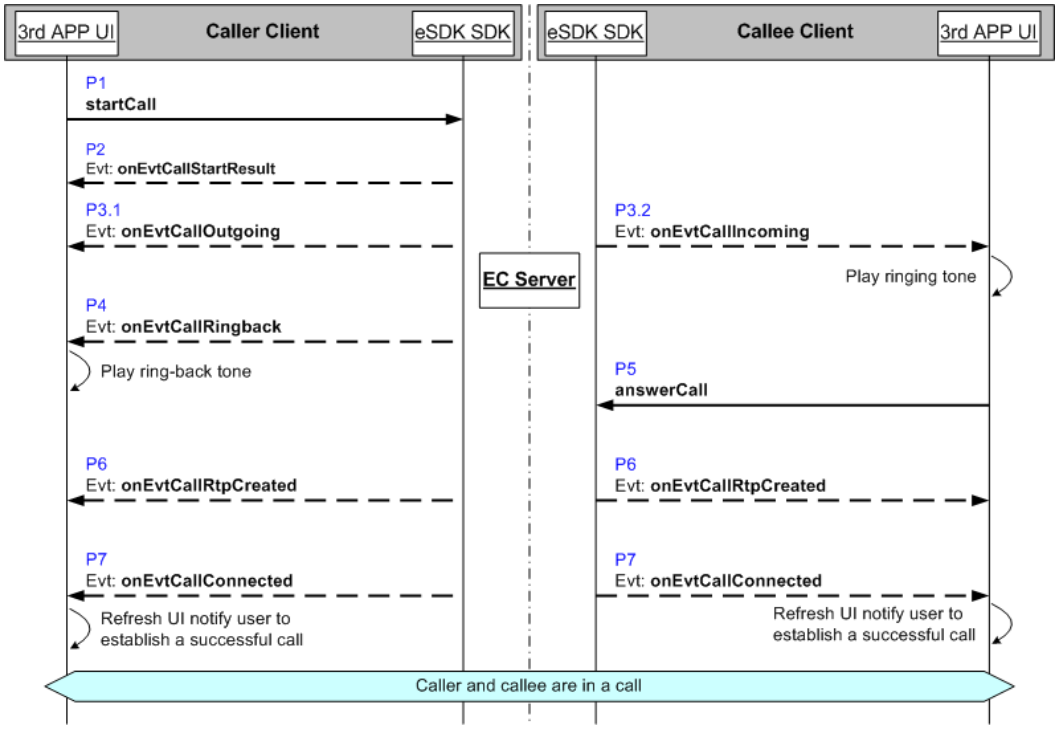
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudEC V600R019C10	
变更	NA	NA	NA

前提条件

主被叫客户端均已注册。

流程说明

图 5-5 建立音频通话流程



步骤1 主叫UI调用TsdkCallManager的startCall()方法发起音频呼叫，传入参数包含被叫号码和是否发起视频呼叫，返回参数为呼叫对象(TsdkCall)。

说明

呼叫ID作为一路通话的唯一标识，UI应保存并管理，以用于后继的呼叫相关操作。

代码示例：

```
//Java code
TsdkCall call = TsdkManager.getInstance().getCallManager().startCall(toNumber, isVideoCall);
```

步骤2 主叫SDK完成状态自检，准备呼叫请求消息，向UI上报呼出事件发起呼叫结果事件onEvtCallStartResult。

步骤3 主叫SDK发出呼叫请求消息，向UI上报呼出事件onEvtCallOutgoing；被叫SDK收到呼叫请求消息，向UI上报来电事件onEvtCallIncoming。通知事件携带参数TsdkCall，包含远端号码、呼叫类型和呼叫状态等呼叫信息。

代码示例：

```
//Java code
public void onEvtCallIncoming(TsdkCall call, Boolean maybeVideoCall){
    Log.i(TAG, "onCallComing");
    if (null == call)
    {
        Log.e(TAG, "onCallComing call is null");
        return;
    }
    Session newSession = new Session(call);
    putCallSessionToMap(newSession);

    CallInfo callInfo = getCallInfo(call);
    callInfo.setMaybeVideoCall(maybeVideoCall);
```

```
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.CALL_COMING, callInfo);
}
```

步骤4 主叫SDK在收到被叫振铃通知时，上报onEvtCallRingback事件，UI应播放本地回铃音。

代码示例：

```
//Java code
//Java code
public void onEvtCallRingback(TsdkCall call){
    Log.i(TAG, "onCallRingBack");
    if (null == call)
    {
        Log.e(TAG, "onCallRingBack call is null");
        return;
    }
        if (null != mCallNotification)
        {
            mCallNotification.onCallEventNotify(CallConstant.CallEvent.PLAY_RING_BACK_TONE, null);
        }
}
```

步骤5 被叫调用TsdkCall的answerCall()方法接听呼叫。

说明

被叫若拒绝呼叫参见**结束通话（或呼叫）**章节描述。

代码示例：

```
//Java code
int result = tsdkCall.answerCall(iVideoCall==1? true:false);
```

步骤6 主被叫SDK向UI上报RTP通道已建立事件onEvtCallRtpCreated。

说明

RTP通道已建立，可以进行一些二次拨号等操作。

代码示例：

```
//Java code
public void onEvtCallRtpCreated(TsdkCall call){
    Log.i(TAG, "onCallRTPCreated");
    if (null == call)
    {
        Log.e(TAG, "tupCall obj is null");
        return;
    }

    CallInfo callInfo = getCallInfo(call);

    mCallNotification.onCallEventNotify(CallConstant.CallEvent.RTP_CREATED, callInfo);
}
```

步骤7 主被叫SDK向UI上报通话建立事件onEvtCallConnected，UI刷新界面进入通话界面，主被叫双方通话。

代码示例：

```
//Java code
public void onEvtCallConnected(TsdkCall call){
    Log.i(TAG, "onCallConnected");
    if (null == call)
    {
        Log.e(TAG, "call obj is null");
        return;
    }

    CallInfo callInfo = getCallInfo(call);
```



```
    Session callSession = getCallSessionByCallID(call.getCallInfo().getCallId());
    if (callSession == null)
    {
        Log.e(TAG, "call session obj is null");
        return;
    }

    if (callInfo.isVideoCall())
    {
        callSession.setCallStatus(CallConstant.CallStatus.VIDEO_CALLING);
    }
    else
    {
        callSession.setCallStatus(CallConstant.CallStatus.AUDIO_CALLING);
    }

    mCallNotification.onCallEventNotify(CallConstant.CallEvent.CALL_CONNECTED, callInfo);
}
```

----结束

注意事项

无。

5.4.3 建立视频通话

应用场景

用户点对点视频通话。

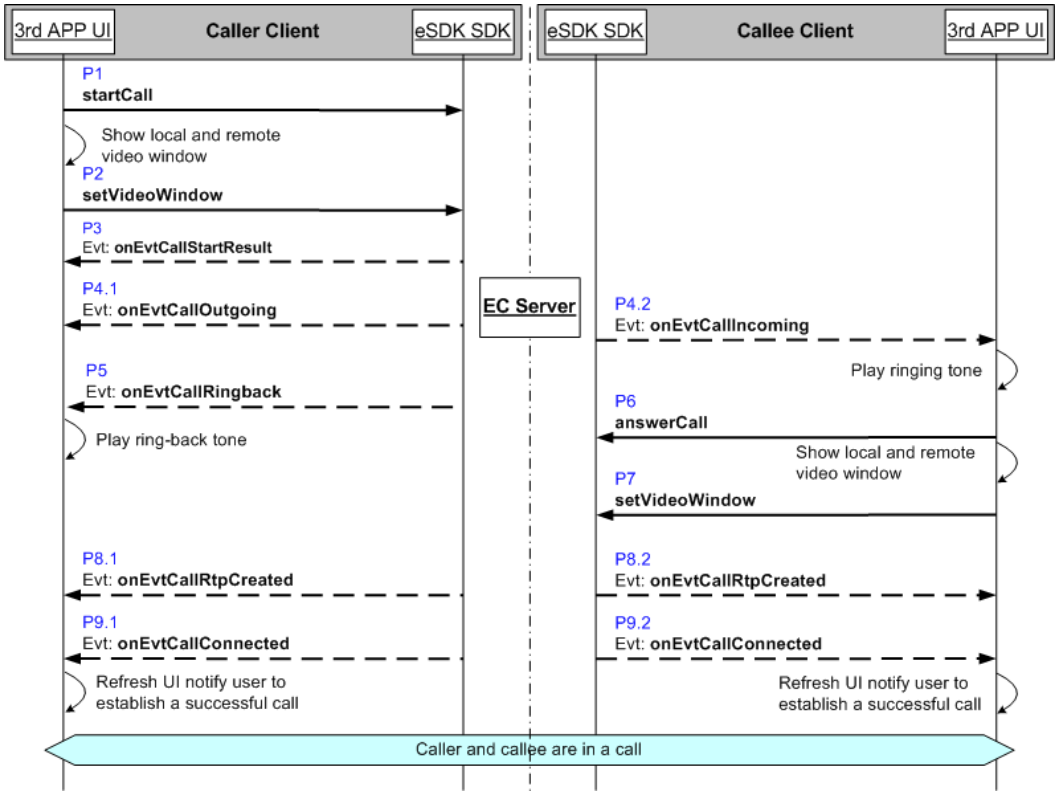
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

主被叫客户端均已注册。

流程说明

图 5-6 建立视频通话流程



步骤1 主叫UI调用TsdkCallManager的startCall()方法发起视频呼叫，传入参数包含被叫号码和是否发起视频呼叫，返回参数为呼叫对象(TsdkCall)。

说明

呼叫ID作为一路通话的唯一标识，UI应保存并管理，以用于后继的呼叫相关操作。

代码示例：

```
//Java code
TsdkCall call = TsdkManager.getInstance().getCallManager().startCall(toNumber, isVideoCall);
```

步骤2 主叫发起视频呼叫后，就调用TsdkCall的setVideoWindow()方法设置视频窗口与呼叫的绑定关系。

代码示例：

```
//Java code
TsdkVideoWndInfo localWndInfo = new TsdkVideoWndInfo();
localWndInfo.setVideoWndType(TsdkVideoWndType.TSDK_E_VIDEO_WND_LOCAL);
localWndInfo.setRender(ViERenderer.getIndexOfSurface(localVideoView));
localWndInfo.setDisplayMode(TsdkVideoWndDisplayMode.TSDK_E_VIDEO_WND_DISPLAY_FULL);

TsdkVideoWndInfo remoteWndInfo = new TsdkVideoWndInfo();
remoteWndInfo.setVideoWndType(TsdkVideoWndType.TSDK_E_VIDEO_WND_REMOTE);
remoteWndInfo.setRender(ViERenderer.getIndexOfSurface(remoteVideoView));
remoteWndInfo.setDisplayMode(TsdkVideoWndDisplayMode.TSDK_E_VIDEO_WND_DISPLAY_CUT);

List<TsdkVideoWndInfo> list = new ArrayList<>();
list.add(localWndInfo);
list.add(remoteWndInfo);
```



```
TsdkManager.getInstance().getCallManager().getCallByCallId(callId).setVideoWindow(list);
```

步骤3 主叫SDK发出呼叫请求消息，向UI上报发起呼叫结果事件onEvtCallStartResult。

步骤4 主叫SDK发出呼叫请求消息，向UI上报呼出事件onEvtCallOutgoing；被叫SDK收到呼叫请求消息，向UI上报来电事件onEvtCallIncoming。通知事件携带参数TsdkCall，包含远端号码、呼叫类型和呼叫状态等呼叫信息。

代码示例：

```
//Java code
public void onEvtCallOutgoing(TsdkCall call){
    Log.i(TAG, "onCallGoing");
    if (null == call)
    {
        Log.e(TAG, "tupCall obj is null");
        return;
    }
    CallInfo callInfo = getCallInfo(call);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.CALL_GOING, callInfo);
}
```

```
//Java code
public void onEvtCallIncoming(TsdkCall call, Boolean maybeVideoCall){
    Log.i(TAG, "onCallComing");
    if (null == call)
    {
        Log.e(TAG, "onCallComing call is null");
        return;
    }
    Session newSession = new Session(call);
    putCallSessionToMap(newSession);

    CallInfo callInfo = getCallInfo(call);
    callInfo.setMaybeVideoCall(maybeVideoCall);

    mCallNotification.onCallEventNotify(CallConstant.CallEvent.CALL_COMING, callInfo);
}
```

步骤5 主叫SDK在收到被叫振铃通知时，上报onEvtCallRingback事件，UI应播放本地回铃音。

代码示例：

```
//Java code
public void onEvtCallRingback(TsdkCall call){
    Log.i(TAG, "onCallRingBack");
    if (null == call)
    {
        Log.e(TAG, "onCallRingBack call is null");
        return;
    }
        if (null != mCallNotification)
        {
            mCallNotification.onCallEventNotify(CallConstant.CallEvent.PLAY_RING_BACK_TONE, null);
        }
}
```

步骤6 被叫调用TsdkCall的answerCall()方法接听呼叫。

说明

被叫若拒绝呼叫参见**结束通话（或呼叫）**章节描述。

代码示例：

```
//Java code
int result = tsdkCall.answerCall(iVideoCall==1? true:false);
```


步骤7 被叫用户若视频接听，UI先完成本地窗口和远端窗口创建，再调用TsdkCall的setVideoWindow()方法设置视频窗口与呼叫的绑定关系。

说明

被叫用户若选择音频接听，则被叫用户无需此步骤。

步骤8 主被叫SDK向UI上报RTP通道已建立事件onEvtCallRtpCreated。

说明

RTP通道已建立，可以进行一些二次拨号等操作。

代码示例：

```
//Java code
public void onEvtCallRtpCreated(TsdkCall call){
    Log.i(TAG, "onCallRTPCreated");
    if (null == call)
    {
        Log.e(TAG, "tupCall obj is null");
        return;
    }

    CallInfo callInfo = getCallInfo(call);

    mCallNotification.onCallEventNotify(CallConstant.CallEvent.RTP_CREATED, callInfo);
}
```

步骤9 主被叫SDK向UI上报通话建立事件onEvtCallConnected，UI刷新界面进入通话界面，主被叫双方通话。

代码示例：

```
//Java code
public void onEvtCallConnected(TsdkCall call){
    Log.i(TAG, "onCallConnected");
    if (null == call)
    {
        Log.e(TAG, "call obj is null");
        return;
    }

    CallInfo callInfo = getCallInfo(call);
    Session callSession = getCallSessionByCallID(call.getCallInfo().getCallId());
    if (callSession == null)
    {
        Log.e(TAG, "call session obj is null");
        return;
    }

    if (callInfo.isVideoCall())
    {
        callSession.setCallStatus(CallConstant.CallStatus.VIDEO_CALLING);
    }
    else
    {
        callSession.setCallStatus(CallConstant.CallStatus.AUDIO_CALLING);
    }

    mCallNotification.onCallEventNotify(CallConstant.CallEvent.CALL_CONNECTED, callInfo);
}
```

----结束

注意事项

无。

eSDK CloudVC 20.1.RC1 开发指南(Android)

5.4.4 结束通话(或呼叫)

应用场景

用户在通话接通前可进行取消呼叫或拒绝呼叫等操作，通话过程中可以结束通话。

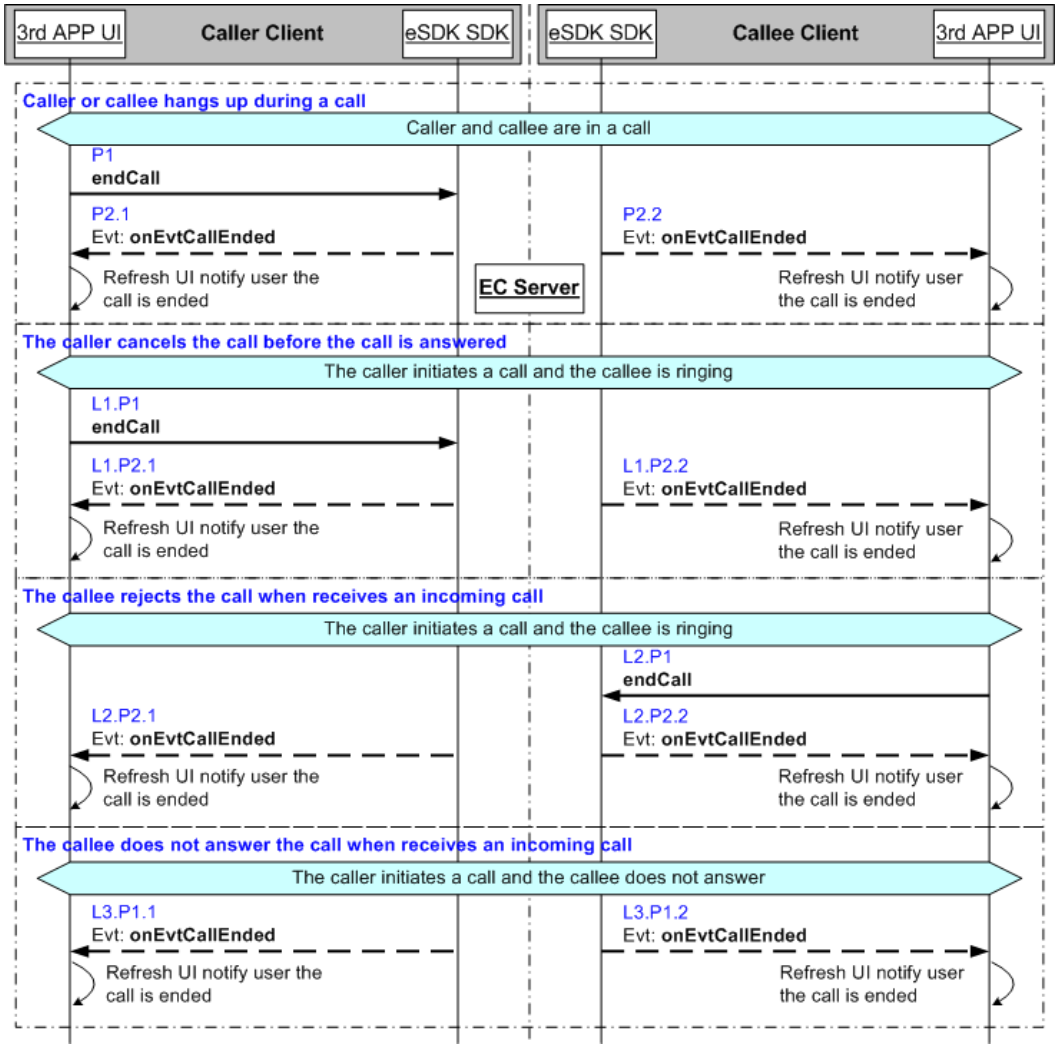
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

详见各子场景说明。

流程说明

图 5-7 结束通话(呼叫)流程



主叫或被叫在通话过程中挂断呼叫

说明

通话双方均可挂断呼叫，本章节以主叫发起挂断为例。

步骤1 主叫UI调用TsdkCall的endCall()方法挂断通话。

代码示例：

```
//Java code
int result = tsdkCall.endCall();
```

步骤2 主、被叫SDK完成呼叫挂断信令交互，向UI上报通话结束事件onEvtCallEnded，UI刷新界面显示通话结束。

代码示例：

```
//Java code
public void onEvtCallEnded(TsdkCall call){
    Log.i(TAG, "onCallEnded");
    if (null == call)
    {
        Log.e(TAG, "onCallEnded call is null");
        return;
    }
    CallInfo callInfo = getCallInfo(call);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.CALL_ENDED, callInfo);
}
```

----结束

主叫在通话接通前取消呼叫

步骤1 主叫UI调用TsdkCall的endCall()方法取消呼叫。

代码示例：

```
//Java code
int result = tsdkCall.endCall();
```

步骤2 主、被叫SDK完成呼叫挂断信令交互，向UI上报通话结束事件onEvtCallEnded，UI刷新界面显示通话结束。

代码示例：

```
//Java code
public void onEvtCallEnded(TsdkCall call){
    Log.i(TAG, "onCallEnded");
    if (null == call)
    {
        Log.e(TAG, "onCallEnded call is null");
        return;
    }
    CallInfo callInfo = getCallInfo(call);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.CALL_ENDED, callInfo);
}
```

----结束

被叫在收到来电时拒绝呼叫

步骤1 被叫UI调用TsdkCall的endCall()方法拒绝通话。

代码示例：

```
//Java code
int result = tsdkCall.endCall();
```


步骤2 主、被叫SDK完成呼叫挂断信令交互，向UI上报通话结束事件onEvtCallEnded，UI刷新界面显示通话结束。

说明

在实际现网部署过程中，业务服务器可能开启了呼叫等待功能，在被叫拒绝呼叫时，业务服务器会与主叫建立通话并播放提示音“对方忙”或“对方通话中”，主叫SDK在此过程中会上报onEvtCallConnected事件，需要用户通过主叫UI主动挂断呼叫。

代码示例：

```
//Java code
public void onEvtCallEnded(TsdkCall call){
    Log.i(TAG, "onCallEnded");
    if (null == call)
    {
        Log.e(TAG, "onCallEnded call is null");
        return;
    }
    CallInfo callInfo = getCallInfo(call);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.CALL_ENDED, callInfo);
}
```

----结束

被叫在收到来电时未接听

步骤1 主、被叫SDK向UI上报通话结束事件onEvtCallEnded，UI刷新界面显示通话结束。

说明

在实际现网部署过程中，业务服务器可能开启了呼叫等待功能，在被叫拒绝呼叫时，业务服务器会与主叫建立通话并播放提示音“对方未接听”，主叫SDK在此过程中会上报onEvtCallRtpCreated或onEvtCallConnected事件，需要用户通过主叫UI主动挂断呼叫。

代码示例：

```
//Java code
public void onEvtCallEnded(TsdkCall call){
    Log.i(TAG, "onCallEnded");
    if (null == call)
    {
        Log.e(TAG, "onCallEnded call is null");
        return;
    }
    CallInfo callInfo = getCallInfo(call);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.CALL_ENDED, callInfo);
}
```

----结束

注意事项

无。

5.4.5 获取呼叫信息

应用场景

获取呼叫信息。

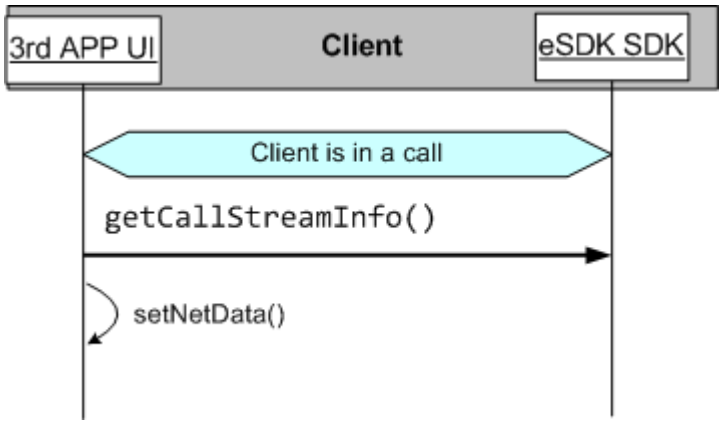
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	LVC V100R019C10	
变更	NA	NA	NA

前提条件

1. 鉴权登录成功；
2. 呼叫建立或者会议创建成功；

流程说明

图 5-8 获取呼叫信息流程



步骤1 UI调用接口getCallStreamInfo()获取呼叫信息，参数为呼叫标识callid和需要获取的Qos信息。

说明

UI定时5s调用此接口信息，刷新呼叫信息，并显示在网络状态界面信息中。

代码示例：

```
TsdkCallStreamInfo    mQosInfo = mQosInfo =
TsdkManager.getInstance().getCallManager().getCallStreamInfo(callid)
音频质量    mQosInfo.getAudioStreamInfo().getRecvJitter()
            ...
视频质量    mQosInfo.getVideoStreamInfo().getRecvBytes()
            ...
```

----结束

注意事项

无。

5.4.6 二次拨号

应用场景

一些业务场景中，需要用户通过终端按键与网络进行交互，如充值、拨打总机后再拨打分机号码、拨打客服中心号码等，二次拨号功能，即DTMF（Dual Tone Multi-Frequency）功能就是为了满足这种需求而产生的。拨打电信运营商的号码后，收到提示音需要进行按键操作时，也是通过该功能完成。

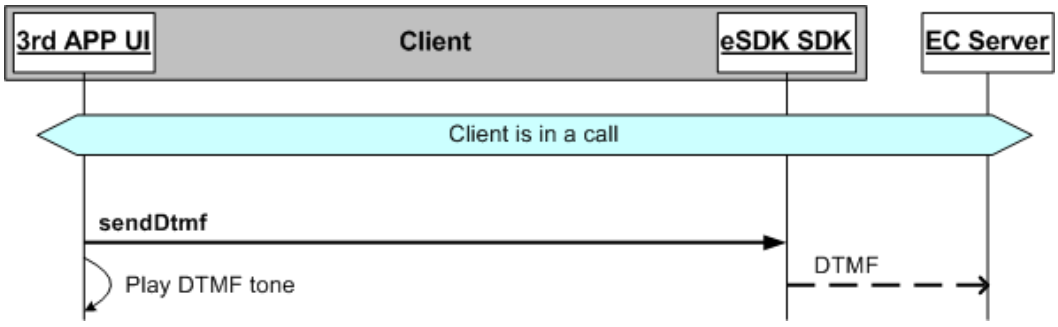
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK EC 6.1.0	CloudEC V600R006C10SPC300	
变更	NA	NA	NA

前提条件

已建立与业务服务器间的通话。

流程说明

图 5-9 二次拨号流程



步骤1 UI调用TsdkCall对象中的sendDtmf()方法在通话中发送DTMF信号，参数为DTMF键值取值0到9，*为10,#为11；UI需要在通话界面上提供一个标准拨号盘，根据用户的输入，发送对应的DTMF信号。

说明

SDK不提供DTMF按键音功能。为了实现更友好的最终用户体验，UI应同步调用SDK提供的媒体播放接口或系统提供的播放接口，实现播放DTMF按键音。

代码示例：

```
//Java code
public boolean reDial(int code)
{
    TsdkDtmfTone tsdkDtmfTone = TsdkDtmfTone.enumOf(code);
    LogUtil.d(TAG, "Dtmf Tone ： " + tsdkDtmfTone.getIndex());
    int result = tsdkCall.sendDtmf(tsdkDtmfTone);
    if (result != 0)
```



```
    {
        LogUtil.e(TAG, "sendDTMF return failed, result = " + result);
        return false;
    }
    return true;
}
```

----结束

注意事项

无。

5.4.7 音频通话转视频通话

应用场景

音频通话中，通话的一方发起音频通话切换为视频通话。

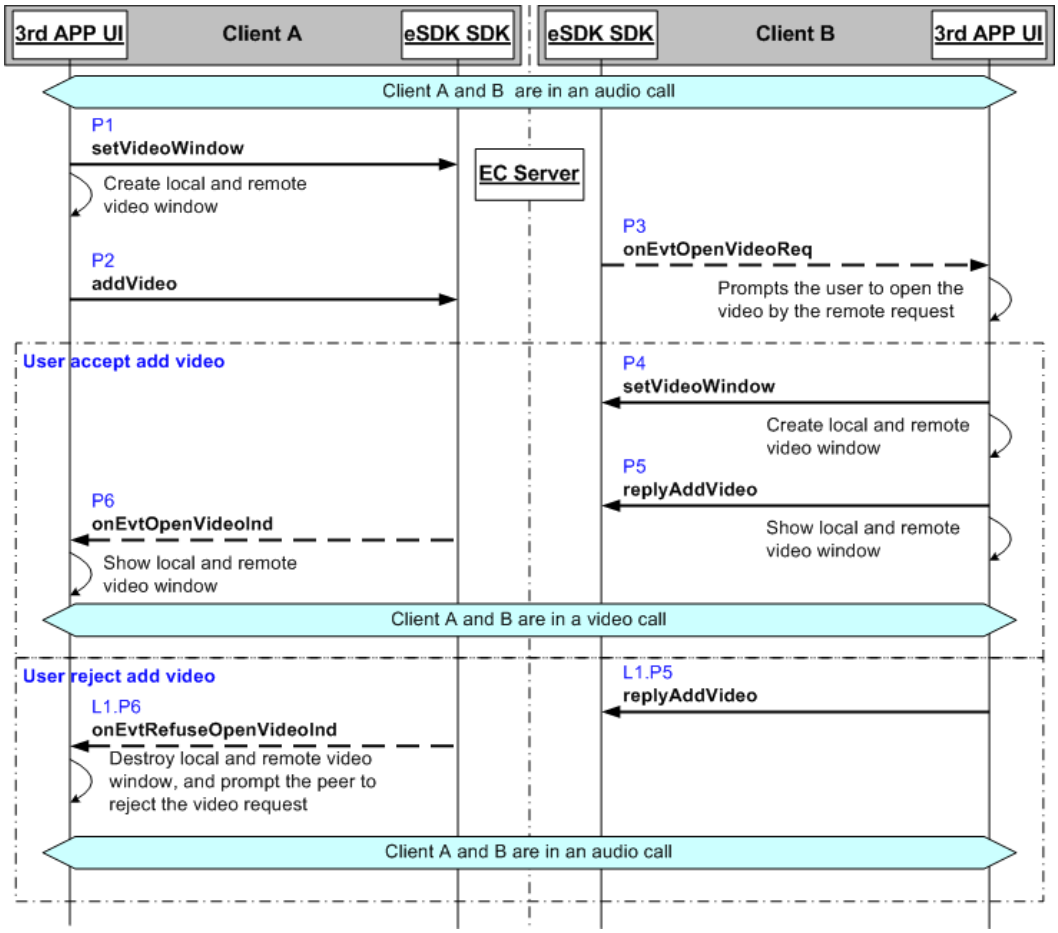
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

音频通话已建立，主被叫正在通话中。

流程说明

图 5-10 音频通话转视频通话流程



说明

通话中，主被叫双方均可以发起音频转视频操作。

步骤1 请求发起方UI先完成本地窗口和远端窗口创建，再调用TsdkCall对象中的setVideoWindow()方法设置视频窗口信息。

代码示例：

```
//Java code
TsdkVideoWndInfo localWndInfo = new TsdkVideoWndInfo();
localWndInfo.setVideoWndType(TsdkVideoWndType.TSDK_E_VIDEO_WND_LOCAL);
localWndInfo.setRender(ViERenderer.getIndexOfSurface(localVideoView));
localWndInfo.setDisplayMode(TsdkVideoWndDisplayMode.TSDK_E_VIDEO_WND_DISPLAY_FULL);

TsdkVideoWndInfo remoteWndInfo = new TsdkVideoWndInfo();
remoteWndInfo.setVideoWndType(TsdkVideoWndType.TSDK_E_VIDEO_WND_REMOTE);
remoteWndInfo.setRender(ViERenderer.getIndexOfSurface(remoteVideoView));
remoteWndInfo.setDisplayMode(TsdkVideoWndDisplayMode.TSDK_E_VIDEO_WND_DISPLAY_CUT);

List<TsdkVideoWndInfo> list = new ArrayList<>();
list.add(localWndInfo);
list.add(remoteWndInfo);

TsdkManager.getInstance().getCallManager().getCallByCallId(callId).setVideoWindow(list);
```


步骤2 请求发起方UI调用TsdkCall对象中的addVideo()方法发起音频转视频呼叫请求。

代码示例：

```
//Java code
public boolean addVideo()
{
    initVideoWindow();

    int result = tsdkCall.addVideo();
    if (result != 0)
    {
        LogUtil.e(TAG, "addVideo return failed, result = " + result);
        return false;
    }

    setCallStatus(CallConstant.CallStatus.VIDEO_CALLING);
    return true;
}
```

步骤3 被请求方SDK收到请求后，通过TsdkNotify对象中的onEvtOpenVideoReq()方法向UI上报对方请求音频转视频事件，UI应刷新界面通知用户远端请求转视频。

代码示例：

```
//Java code
public void onEvtOpenVideoReq(TsdkCall call, TsdkVideoOrientation orientType){
    Log.i(TAG, "onCallAddVideo");
    if (null == call)
    {
        Log.e(TAG, "onCallAddVideo tupCall is null");
        return;
    }
    Session callSession = getCallSessionByCallID(call.getCallInfo().getCallId());
    if (callSession == null)
    {
        Log.e(TAG, "call session obj is null");
        return;
    }

    CallConstant.CallStatus callStatus = callSession.getCallStatus();
    boolean isSupportVideo = isSupportVideo();

    if ((!isSupportVideo) || (CallConstant.CallStatus.AUDIO_CALLING != callStatus))
    {
        callSession.rejectAddVideo();
        return;
    }

    mCallNotification.onCallEventNotify(CallConstant.CallEvent.RECEIVED_REMOTE_ADD_VIDEO_REQUEST,
null);

}
```

步骤4 被请求方接受转视频请求，UI先完成本地窗口和远端窗口创建，再调用TsdkCall对象中的setVideoWindow()方法设置视频窗口信息。

说明

只有被请求方用户接受时，才需要此步骤。

若用户长时间没有响应，被请求端应用程序应该自动拒绝转视频的请求。建议时间为45s。

步骤5 被请求方UI调用TsdkCall对象中的replyAddVideo()方法接受转视频请求。

说明

参数 isAccept表示是否接受: true为同意，false为拒绝。

代码示例：

eSDK CloudVC 20.1.RC1 开发指南(Android)


```
//Java code
public boolean acceptAddVideo()
{
    initVideoWindow();

    int result = tsdkCall.replyAddVideo(true);
    if (result != 0)
    {
        LogUtil.e(TAG, "replyAddVideo(accept) return failed, result = " + result);
        return false;
    }
    return true;
}
```

步骤6 主、被叫SDK完成视频转音频信令和媒体交互处理，若被请求方接受视频请求，主叫SDK通过TsdkNotify对象中的onEvtOpenVideoInd()方法向UI上报打开视频通知事件，主叫UI根据事件显示远端和近端视频窗口；被请求方点击接受后，被叫UI显示远端和近端视频窗口。若被请求方拒绝视频请求，主叫SDK通过TsdkNotify对象中的onEvtRefuseOpenVideoInd()方法向UI上报远端拒绝请求打开视频通知事件，主叫UI销毁远端和近端视频窗口。

代码示例：

```
//Java code
public void onEvtRefuseOpenVideoInd(TsdkCall call){

    VideoMgr.getInstance().clearCallVideo();

    Session callSession = getCallSessionByCallID(call.getCallInfo().getCallId());
    callSession.setCallStatus(CallConstant.CallStatus.AUDIO_CALLING);

    CallInfo callInfo = getCallInfo(call);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.REMOTE_REFUSE_ADD_VIDEO_SREQUEST,
callInfo);

}
```

```
//Java code
public void onEvtOpenVideoInd(TsdkCall call){
    int isVideo = call.getCallInfo().getIsVideoCall(); // 1:video, 0: audio
    int callId  = call.getCallInfo().getCallId();
    Log.i(TAG,  "isVideo: " + isVideo + "callId: " + callId);

    Session callSession = getCallSessionByCallID(callId);
    if (callSession == null)
    {
        return;
    }
    CallInfo callInfo = getCallInfo(call);//audio --> video success
    Log.i(TAG, "Upgrade To Video Call");
    VideoMgr.getInstance().setVideoOrient(callId, CallConstant.FRONT_CAMERA);

    callSession.setCallStatus(CallConstant.CallStatus.VIDEO_CALLING);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.OPEN_VIDEO, callInfo);
}
```

----结束

注意事项

无。

5.4.8 视频通话转音频通话

应用场景

视频通话中，通话的一方发起视频通话切换为音频通话。

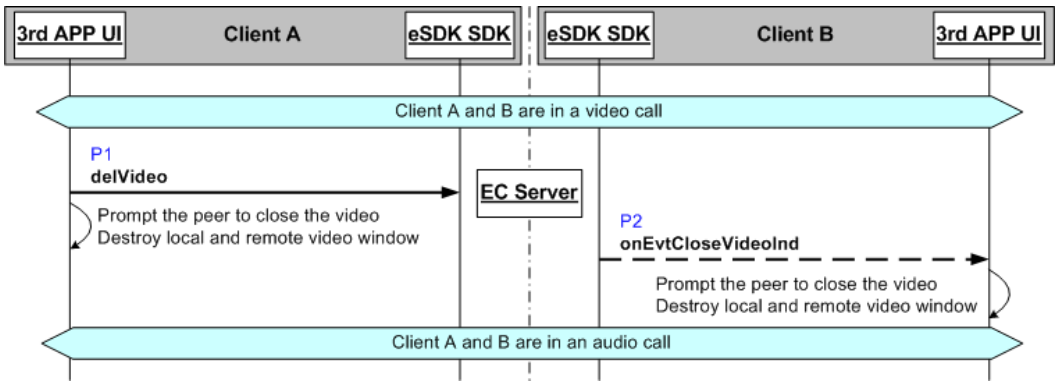
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

视频通话已建立，主被叫正在通话中。

流程说明

图 5-11 视频通话转音频通话流程



说明

通话中，主被叫双方均可以发起视频转音频操作。

步骤1 请求发起方UI调用TsdkCall对象中的delVideo()方法发起视频转音频呼叫请求。

代码示例：

```
//Java code
public boolean delVideo()
{
    int result = tsdkCall.delVideo();
    if (result != 0)
    {
        LogUtil.e(TAG, "delVideo return failed, result = " + result);
        return false;
    }

    setCallStatus(CallConstant.CallStatus.AUDIO_CALLING);

    return true;
}
```

步骤2 被请求方SDK收到请求后，自动进行视频转音频交互处理，通过TsdkNotify对象中的onEvtCloseVideoInd()方法向UI上报对方请求关闭视频通知，UI刷新界面提示用户关闭摄像头，销毁远端和近端视频窗口，刷新界面，无需用户确认。

代码示例：

```
//Java code
public void onEvtCloseVideoInd(TsdkCall call){
    if (null == call)
    {
        Log.e(TAG, "onCallDelVideo tupCall is null");
        return;
    }

    Session callSession = getCallSessionByCallID(call.getCallInfo().getCallId());
    if (callSession == null)
    {
        Log.e(TAG, "call session obj is null");
        return;
    }

    callSession.setCallStatus(CallConstant.CallStatus.AUDIO_CALLING);

    //Clear video data
    VideoMgr.getInstance().clearCallVideo();

    if (null != mCallNotification)
    {
        CallInfo callInfo = getCallInfo(call);
        mCallNotification.onCallEventNotify(CallConstant.CallEvent.CLOSE_VIDEO, callInfo);
    }

    if (callSession.isVideoHold())
    {
        callSession.holdCall();
    }
}
```

----结束

注意事项

无。

5.4.9 保持和恢复音频通话

应用场景

用户可以将正在进行的音频通话保持，待需要的时候再恢复通话，能减少拨号次数，使用更方便。

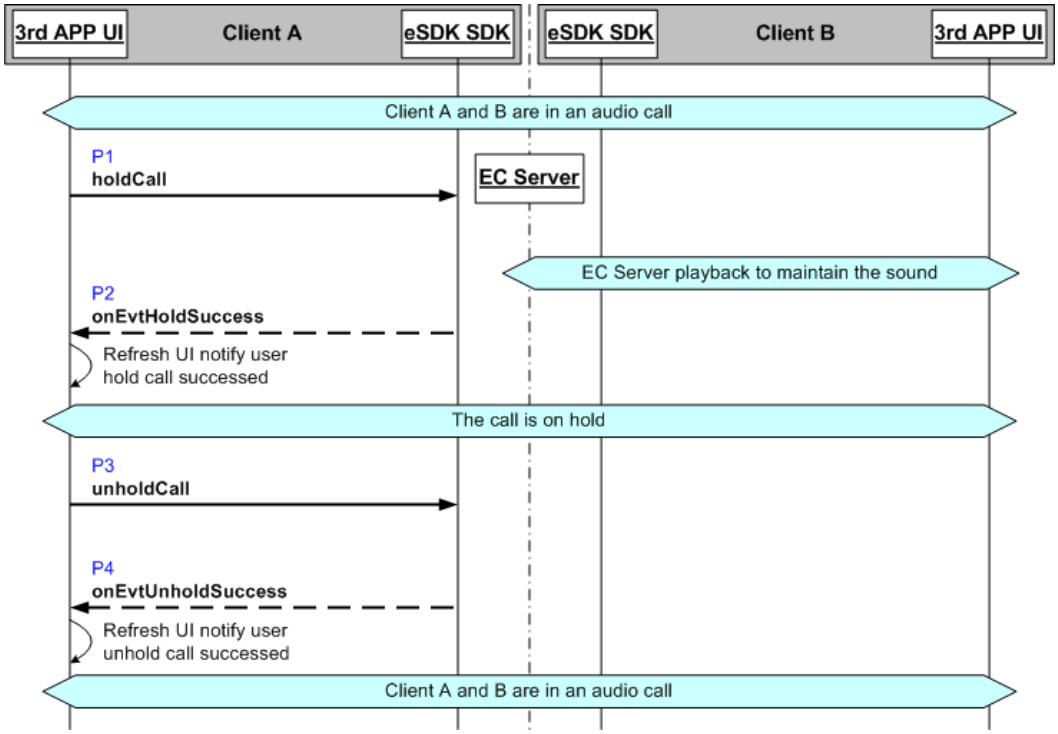
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

音频通话已建立，主被叫正在通话中。

流程说明

图 5-12 保持和恢复音频通话流程



保持通话

说明

通话中，主被叫双方均可以发起保持通话操作。

步骤1 请求发起方UI调用TsdkCall对象中的holdCall()方法保持通话。

代码示例：

```
//Java code
public boolean holdCall()
{
    int result = tsdkCall.holdCall();
    if (result != 0)
    {
        LogUtil.e(TAG, "holdCall return failed, result = " + result);
        return false;
    }
    return true;
}
```

步骤2 请求方SDK发送保持通话请求，与业务服务器和被请求方完成通话保持交互处理，请求方SDK通过TsdkNotify对象中的onEvtHoldSuccess()方法向UI上报保持通话成功事件， 呼叫状态为保持，UI刷新界面提示当前通话保持中，并显示恢复通话入口；被请求方UI界面不感知当前通话被保持，依然处于通话态。

说明

1、保持通话可能由于服务器权限或当前通话业务限制，导致用户保持通话失败，此时SDK通过TsdkNotify对象中的onEvtHoldFailed()方法向UI上报保持通话失败事件，通话状态为通话中；

2、在业务服务器支持播放保持提示音时，被保持方可以听到由服务器侧播放的保持提示音；

3、为了较优的业务体验，建议在通话保持时，UI屏蔽挂断通话入口。

代码示例：

```
//Java code
public void onEvtHoldFailed(TsdkCall call){
    Log.i(TAG, "handleHoldFailed");
    CallInfo callInfo = getCallInfo(call);
    Session callSession = getCallSessionByCallID(callInfo.getCallID());
    if (callSession.isVideoHold())
    {
        callSession.setVideoHold(false);
        //保持失败，只直接通知UI失败，不自动动恢复视频
        mCallNotification.onCallEventNotify(CallConstant.CallEvent.VIDEO_HOLD_FAILED, callInfo);
    }
    else
    {
        mCallNotification.onCallEventNotify(CallConstant.CallEvent.AUDIO_HOLD_FAILED, callInfo);
    }
}
```

```
//Java code
public void onEvtHoldSuccess(TsdkCall call){
    Log.i(TAG, "handleHoldSuccess");
    CallInfo callInfo = getCallInfo(call);
    Session callSession = getCallSessionByCallID(callInfo.getCallID());
    if (callSession.isVideoHold())
    {
        mCallNotification.onCallEventNotify(CallConstant.CallEvent.VIDEO_HOLD_SUCCESS, callInfo);
    }
    else
    {
        mCallNotification.onCallEventNotify(CallConstant.CallEvent.AUDIO_HOLD_SUCCESS, callInfo);
    }
}
```

恢复通话

说明

保持通话发起方才可进行恢复通话操作。

步骤3 请求发起方UI调用TsdkCall对象中的unholdCall()方法恢复处于保持态的通话。

代码示例：

```
//Java code
public boolean unHoldCall()
{
    int result = tsdkCall.unholdCall();
    if (result != 0)
    {
        LogUtil.e(TAG, "unholdCall return failed, result = " + result);
        return false;
    }
    return true;
}
```

步骤4 请求方SDK发送恢复通话请求，与业务服务器和被请求方完成通话恢复交互处理，请求方SDK通过TsdkNotify对象中的onEvtUnholdSuccess()方法向UI上报恢复通话成功事件，呼叫状态为“通话中”，UI刷新界面显示通话中。

说明

恢复通话可能由于网络侧极低概率的冲突或异常，导致用户恢复通话失败，此时SDK通过TsdkNotify对象中的onEvtUnholdFailed() 向UI上报取消保持通话失败事件，并自动挂断通话。

代码示例：

```
//Java code
public void onEvtUnholdSuccess(TsdkCall call){
    Log.i(TAG, "handleUnholdSuccess");
    int callId = call.getCallInfo().getCallId();
    Session callSession = getCallSessionByCallID(callId);
    if (callSession == null)
    {
        Log.e(TAG, "call session obj is null");
        return;
    }

    if (callSession.isVideoHold())
    {
        addVideo(callId);
        callSession.setVideoHold(false);
    }

    CallInfo callInfo = getCallInfo(call);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.UN_HOLD_SUCCESS, callInfo);
}
```

```
//Java code
public void onEvtUnholdFailed(TsdkCall call){
    Log.i(TAG, "handleUnholdFailed");

    CallInfo callInfo = getCallInfo(call);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.UN_HOLD_FAILED, callInfo);
}
```

----结束

注意事项

因通话双方均可以在本端处于通话态发起保持操作，即通话可能会处于双向保持态，在任一保持方发起恢复通话时，仅能恢复本端通话状态，不会恢复对端的通话状态。

5.4.10 保持和恢复视频通话

应用场景

用户可以将正在进行的视频通话保持，待需要的时候再恢复通话，能减少拨号次数，使用更方便。

说明

保持和恢复视频通话，与“保持和恢复音频通话”流程基本相同，不同在于通话被保持时，视频将会自动被关闭，恢复时，视频将会自动打开。

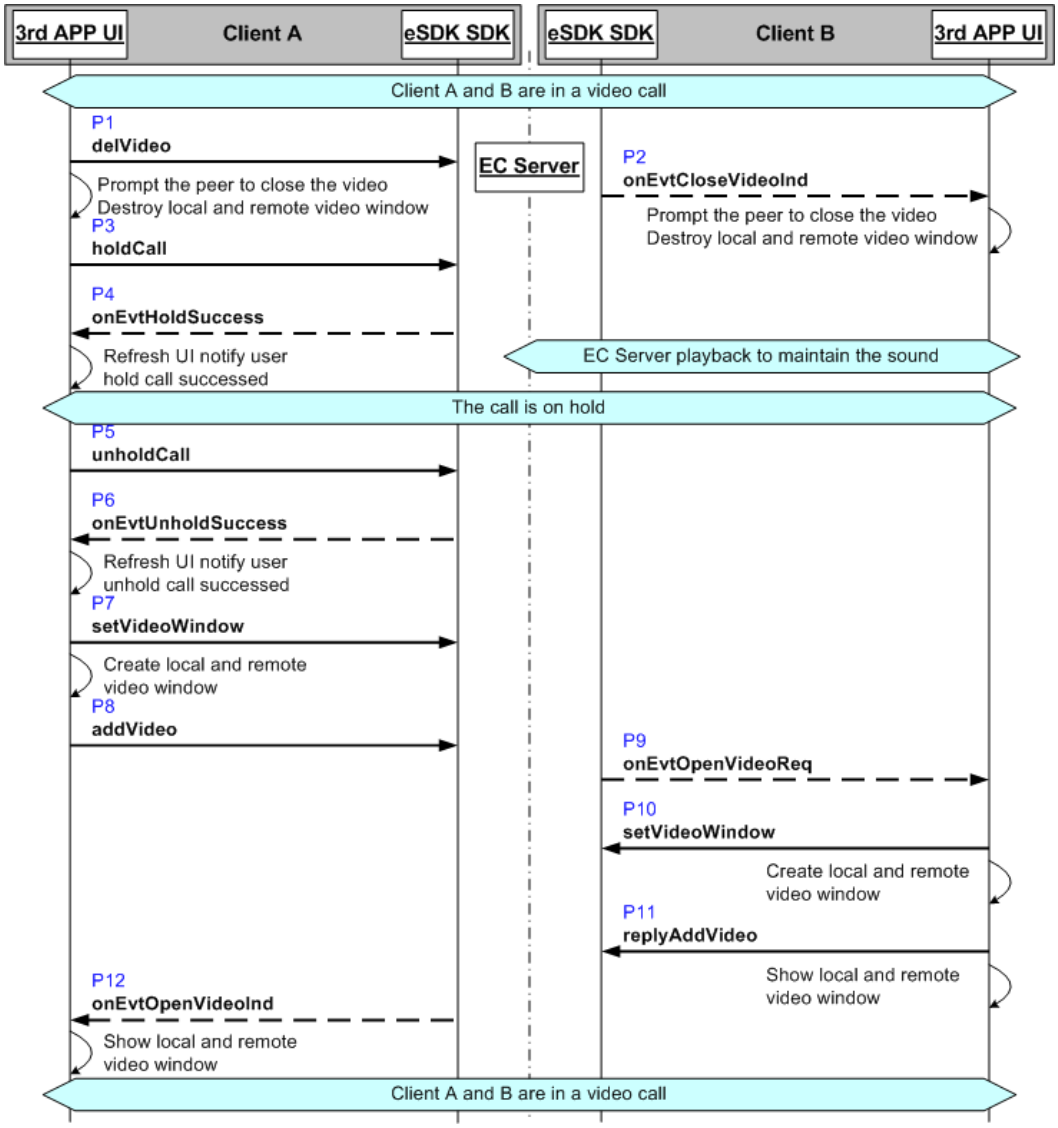
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

视频通话已建立，主被叫正在通话中。

流程说明

图 5-13 保持和恢复视频通话流程



保持通话

说明

通话中，主被叫双方均可以发起保持通话操作。

步骤1 请求发起方UI调用TsdkCall对象中的delVideo()方法发起视频转音频呼叫请求，移除本地和远端视频窗口，UI界面显示为音频通话界面。

说明

当前服务器暂不支持视频通话保持，需要先移除视频窗口，转换为语音通话后再保持通话。

代码示例：

```
//Java code
public boolean delVideo()
{
    int result = tsdkCall.delVideo();
    if (result != 0)
    {
        LogUtil.e(TAG, "delVideo return failed, result = " + result);
        return false;
    }

    setCallStatus(CallConstant.CallStatus.AUDIO_CALLING);

    return true;
}
```

步骤2 被请求方SDK收到请求后，自动进行视频转音频交互处理，通过TsdkNotify对象中的onEvtCloseVideoInd()方法向UI上报对方请求关闭视频通知，移除本地和远端视频窗口，UI界面显示为音频通话界面。

代码示例：

```
//Java code
public void onEvtCloseVideoInd(TsdkCall call){
    if (null == call)
    {
        Log.e(TAG, "onCallDelVideo tupCall is null");
        return;
    }

    Session callSession = getCallSessionByCallID(call.getCallInfo().getCallId());
    if (callSession == null)
    {
        Log.e(TAG, "call session obj is null");
        return;
    }

    callSession.setCallStatus(CallConstant.CallStatus.AUDIO_CALLING);

    //Clear video data
    VideoMgr.getInstance().clearCallVideo();

    if (null != mCallNotification)
    {
        CallInfo callInfo = getCallInfo(call);
        mCallNotification.onCallEventNotify(CallConstant.CallEvent.CLOSE_VIDEO, callInfo);
    }

    if (callSession.isVideoHold())
    {
        callSession.holdCall();
    }
}
```

步骤3 请求发起方UI调用TsdkCall对象中的holdCall()方法保持通话。

步骤4 请求方SDK发送保持通话请求，与业务服务器和被请求方完成通话保持交互处理，请求方SDK通过TsdkNotify对象中的onEvtHoldSuccess()方法向UI上报保持通话成功事件， 呼叫状态为保持，UI刷新界面提示当前通话保持中，并显示恢复通话入口；被请求方UI界面不感知当前通话被保持，依然处于通话态。

说明

1、保持通话可能由于服务器权限或当前通话业务限制，导致用户保持通话失败，此时SDK通过TsdkNotify对象中的onEvtHoldFailed()方法向UI上报保持通话失败事件，通话状态为通话中；

2、在业务服务器支持播放保持提示音时，被保持方可以听到由服务器侧播放的保持提示音；

3、为了较优的业务体验，建议在通话保持时，UI屏蔽挂断通话入口。

恢复通话

说明

保持通话发起方才可进行恢复通话操作。

步骤5 请求发起方UI调用TsdkCall对象中的unholdCall()方法恢复处于保持态的通话。

步骤6 请求方SDK发送恢复通话请求，与业务服务器和被请求方完成通话恢复交互处理，请求方SDK通过TsdkNotify对象中的onEvtUnholdSuccess()方法向UI上报恢复通话成功事件，呼叫状态为“通话中”，UI刷新界面显示通话中。

说明

恢复通话可能由于网络侧极低概率的冲突或异常，导致用户恢复通话失败，此时SDK通过TsdkNotify对象中的onEvtUnholdFailed() 向UI上报取消保持通话失败事件，并自动挂断通话。

步骤7 请求发起方UI先完成本地窗口和远端窗口创建，再调用TsdkCall对象中的setVideoWindow()方法设置视频窗口信息。

代码示例：

```
//Java code
TsdkVideoWndInfo localWndInfo = new TsdkVideoWndInfo();
localWndInfo.setVideoWndType(TsdkVideoWndType.TSDK_E_VIDEO_WND_LOCAL);
localWndInfo.setRender(ViERenderer.getIndexOfSurface(localVideoView));
localWndInfo.setDisplayMode(TsdkVideoWndDisplayMode.TSDK_E_VIDEO_WND_DISPLAY_FULL);

TsdkVideoWndInfo remoteWndInfo = new TsdkVideoWndInfo();
remoteWndInfo.setVideoWndType(TsdkVideoWndType.TSDK_E_VIDEO_WND_REMOTE);
remoteWndInfo.setRender(ViERenderer.getIndexOfSurface(remoteVideoView));
remoteWndInfo.setDisplayMode(TsdkVideoWndDisplayMode.TSDK_E_VIDEO_WND_DISPLAY_CUT);

List<TsdkVideoWndInfo> list = new ArrayList<>();
list.add(localWndInfo);
list.add(remoteWndInfo);

TsdkManager.getInstance().getCallManager().getCallByCallId(callId).setVideoWindow(list);
```

步骤8 请求发起方UI调用TsdkCall对象中的addVideo()方法发起音频转视频呼叫请求。

代码示例：

```
//Java code
public boolean addVideo()
{
    initVideoWindow();

    int result = tsdkCall.addVideo();
    if (result != 0)
    {
        LogUtil.e(TAG, "addVideo return failed, result = " + result);
        return false;
    }

    setCallStatus(CallConstant.CallStatus.VIDEO_CALLING);
    return true;
}
```

步骤9 被请求方SDK收到请求后，通过TsdkNotify对象中的onEvtOpenVideoReq()方法向UI上报对方请求音频转视频事件，UI应刷新界面通知用户远端请求转视频。

代码示例：

```
//Java code
public void onEvtOpenVideoReq(TsdkCall call, TsdkVideoOrientation orientType){
    Log.i(TAG, "onCallAddVideo");
    if (null == call)
```



```
    {
        Log.e(TAG, "onCallAddVideo tupCall is null");
        return;
    }

    Session callSession = getCallSessionByCallID(call.getCallInfo().getCallId());
    if (callSession == null)
    {
        Log.e(TAG, "call session obj is null");
        return;
    }

    CallConstant.CallStatus callStatus = callSession.getCallStatus();
    boolean isSupportVideo = isSupportVideo();

    if ((!isSupportVideo) || (CallConstant.CallStatus.AUDIO_CALLING != callStatus))
    {
        callSession.rejectAddVideo();
        return;
    }

    mCallNotification.onCallEventNotify(CallConstant.CallEvent.RECEIVED_REMOTE_ADD_VIDEO_REQUEST,
null);

}
```

步骤10 被请求方接受转视频请求，UI先完成本地窗口和远端窗口创建，再调用TsdkCall对象中的setVideoWindow()方法设置视频窗口信息。

说明

只有被请求方用户接受时，才需要此步骤。

若用户长时间没有响应，被请求端应用程序应该自动拒绝转视频的请求。建议时间为45s。

步骤11 被请求方UI调用TsdkCall对象中的replyAddVideo()方法接受转视频请求。

说明

参数 isAccept表示是否接受: true为同意，false为拒绝。

代码示例：

```
//Java code
public boolean acceptAddVideo()
{
    initVideoWindow();

    int result = tsdkCall.replyAddVideo(true);
    if (result != 0)
    {
        LogUtil.e(TAG, "replyAddVideo(accept) return failed, result = " + result);
        return false;
    }
    return true;
}
```

步骤12 主、被叫SDK完成视频转音频信令和媒体交互处理，若被请求方接受视频请求，主叫SDK通过TsdkNotify对象中的onEvtOpenVideoInd()方法向UI上报打开视频通知事件，主叫UI根据事件显示远端和近端视频窗口；被请求方点击接受后，被叫UI显示远端和近端视频窗口。若被请求方拒绝视频请求，主叫SDK通过TsdkNotify对象中的onEvtRefuseOpenVideoInd()方法向UI上报远端拒绝请求打开视频通知事件，主叫UI销毁远端和近端视频窗口。

代码示例：

```
//Java code
public void onEvtOpenVideoInd(TsdkCall call){
```



```
    int isVideo = call.getCallInfo().getIsVideoCall(); // 1:video, 0: audio
    int callId  = call.getCallInfo().getCallId();
    Log.i(TAG,  "isVideo: " + isVideo + "callId: " + callId);

    Session callSession = getCallSessionByCallID(callId);
    if (callSession == null)
    {
        return;
    }
    CallInfo callInfo = getCallInfo(call);//audio --> video success
    Log.i(TAG, "Upgrade To Video Call");
    VideoMgr.getInstance().setVideoOrient(callId, CallConstant.FRONT_CAMERA);

    callSession.setCallStatus(CallConstant.CallStatus.VIDEO_CALLING);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.OPEN_VIDEO, callInfo);
}
```

```
//Java code
public void onEvtRefuseOpenVideoInd(TsdkCall call){

    VideoMgr.getInstance().clearCallVideo();

    Session callSession = getCallSessionByCallID(call.getCallInfo().getCallId());
    callSession.setCallStatus(CallConstant.CallStatus.AUDIO_CALLING);

    CallInfo callInfo = getCallInfo(call);
    mCallNotification.onCallEventNotify(CallConstant.CallEvent.REMOTE_REFUSE_ADD_VIDEO_SREQUEST,
callInfo);

}
```

----结束

注意事项

因通话双方均可以在本端处于通话态发起保持操作，即通话可能会处于双向保持态，在任一保持方发起恢复通话时，仅能恢复本端通话状态，不会恢复对端的通话状态；视频通话双向保持中任一方先恢复通话，不会自动打开视频。

5.4.11 通话中闭音麦克风

应用场景

用户通话中设置或取消闭音麦克风，即关闭或打开麦克风，停止或重启音频输入。

说明

设置和取消闭音麦克风针对指定通话，不是针对设备。

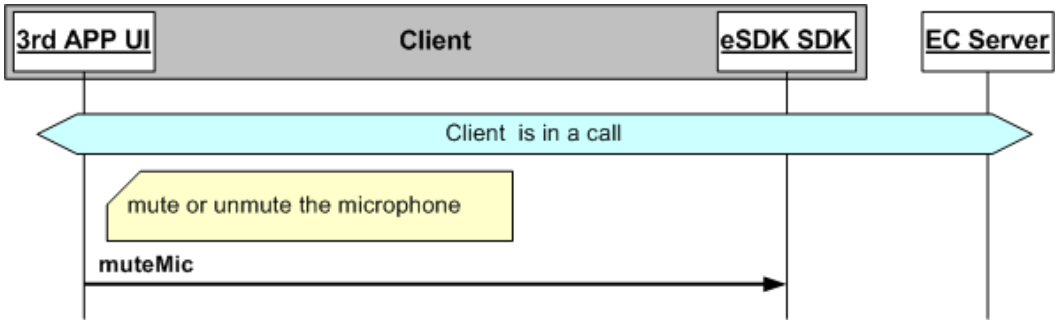
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

通话已建立，主被叫正在通话中。

流程说明

图 5-14 通话中闭音麦克风流程



说明

设置和取消闭音麦克风操作本地媒体，通话对端不感知。

步骤1 UI调用TsdkCall对象中的muteMic()方法关闭或打开麦克风。

说明

参数isMute为是否静音，取值:true为闭音，false为取消闭音。

代码示例：

```
//Java code
public boolean muteMic(boolean mute)
{
    int result = tsdkCall.muteMic(mute);
    if (result != 0)
    {
        LogUtil.e(TAG, "mediaMuteMic return failed, result = " + result);
        return false;
    }
    return true;
}
```

----结束

注意事项

无。

5.4.12 通话中暂停视频

应用场景

用户视频通话中开始或停止视频发送。

说明

设置和取消暂停视频采集针对指定通话，不是针对设备，停止时画面凝固。

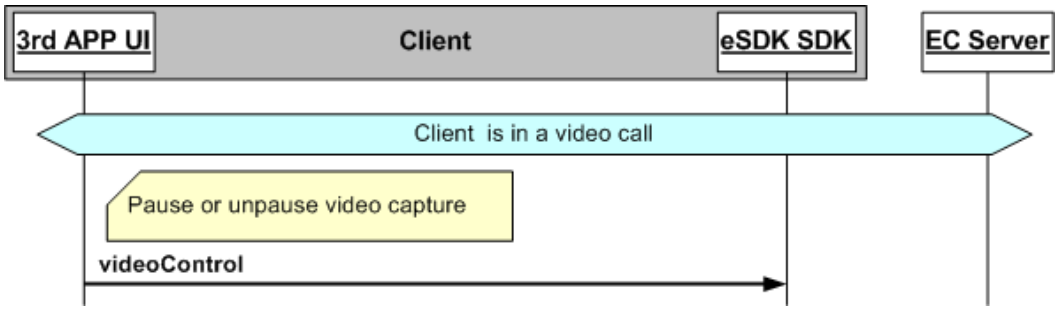
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

视频通话已建立，主被叫正在通话中。

流程说明

图 5-15 通话中暂停视频采集流程



说明

设置和取消暂停视频采集操作本地媒体。

步骤1 UI调用TsdkCall对象中的videoControl()方法暂停或继续视频。

代码示例：

```
//Java code
module = 0x02 | 0x04;
operation = 0x04;

TsdkVideoCtrlInfo tsdkVideoCtrlInfo = new TsdkVideoCtrlInfo(0, operation, module);
result = call.videoControl(tsdkVideoCtrlInfo);
```

----结束

注意事项

无。

5.4.13 设备管理

应用场景

管理音视频设备，包括麦克风、扬声器和摄像头。

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

详见各子场景说明。

流程说明

获取音、视频设备列表

说明

应用程序在任何阶段均可以获取当前可用的音视频设备信息，为更方便地进行后继具体设备管理，建议应用程序在初始化阶段和系统检测到设备变化时，获取设备信息并保存维护。

步骤1 UI调用TsdkCallManager对象中的getDevices()方法获取音频视频设备列表，需要传入的参数为deviceType设备类型。

说明

1.获取麦克风设备列表，请将deviceType设置为TsdkDeviceType.TSDK_E_DEVICE_MIC。

2.获取扬声器设备列表，请将deviceType设置为TsdkDeviceType.TSDK_E_DEVICE_SPEAKER。

3.获取摄像头设备列表，请将deviceType设置为TsdkDeviceType.TSDK_E_DEVICE_CAMERA。

代码示例：

```
//Java code
cameraList =
TsdkManager.getInstance().getCallManager().getDevices(TsdkDeviceType.TSDK_E_DEVICE_CAMERA);
```

----结束

管理音频设备

说明

一般用于用户对音频设备(麦克风和扬声器)进行设置和切换。

步骤1 UI调用TsdkCallManager对象中的setMobileAudioRoute()设置移动音频路由设备。

说明

移动端的音频设备包括：蓝牙、扬声器、听筒和耳机。

代码示例：

```
//java code
private boolean setAudioRoute(TsdkMobileAuidoRoute audioSwitch)
{
    return TsdkManager.getInstance().getCallManager().setMobileAudioRoute(audioSwitch) == 0;
}
```

步骤2 UI调用TsdkCallManager对象中的getMobileAudioRoute()获取移动音频路由设备。

代码示例：

```
//java code
public int getCurrentAudioRoute()
{
    return TsdkManager.getInstance().getCallManager().getMobileAudioRoute().getIndex();
}
```

----结束

管理视频设备

说明

一般用于用户对视频设备进行设置和切换。

步骤1 UI调用TsdkCall对象中的setVideoOrient()设置视频方向。需要传入的参数包括视频设备(摄相头)索引以及视频方向(横竖屏)。

代码示例：

```
//Java code
if (cameraIndex == CallConstant.FRONT_CAMERA) {
    portrait = 3;
    landscape = 0;
    seascape = 2;
} else if (cameraIndex == CallConstant.BACK_CAMERA) {
    portrait = 1;
    landscape = 0;
    seascape = 2;
}

TsdkVideoOrient videoOrient = new TsdkVideoOrient(portrait, seascape, landscape,orient);
int result = callManager.getCallByCallId(callId).setVideoOrient(cameraIndex, videoOrient);
```

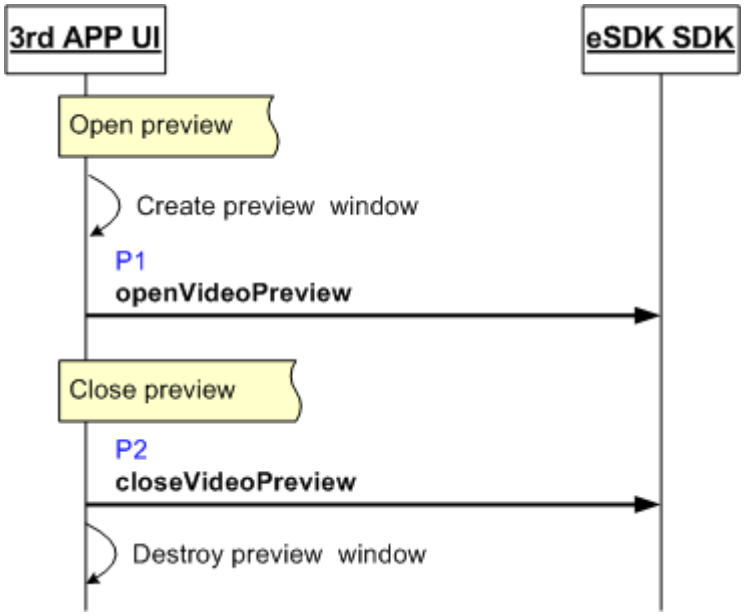
----结束

预览本地视频

说明

一般用于设备设置时，检测本地摄像头工作状态是否正常。

图 5-16 预览本地视频流程



步骤1 UI先创建本地预览窗口，再调用TsdkCallManager对象中的openVideoPreview()方法打开本地视频预览窗口，其中摄像头索引填写“**获取音、视频设备列表**”过程中获取到的摄像头索引。

步骤2 UI调用TsdkCallManager对象中的closeVideoPreview()方法关闭本地视频预览窗口，同时销毁本地预览窗口。

----结束

注意事项

无。

5.5 会议

5.5.1 概述

SDK基于CloudVC解决方案，向第三方开发者开放会议能力。

须知

相对于CloudVC 6.1.0 及之前版本，当前版本及后继版本SDK支持更高效实时的会议控制能力：

1. 集成时，可通过初始化前“设置会议控制参数“，选择会议控制协议(TSDK_E_CONF_CTRL_PROTOCOL)为TSDK_E_CONF_CTRL_PROTOCOL_IDO，启用此能力；
2. 为保持对历史版本的兼容，此能力默认关闭，即使用老版本会议控制协议：TSDK_E_CONF_CTRL_PROTOCOL_REST。
3. 相对老版本会议控制协议：TSDK_E_CONF_CTRL_PROTOCOL_REST，新版本会议控制协议：TSDK_E_CONF_CTRL_PROTOCOL_IDO的会议控制能力存在部分差异，具体参见”会议控制“相关章节描述。

5.5.2 会议管理

5.5.2.1 创建预约会议

应用场景

用户创建预约会议。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	N	

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	eSDK VC 19.1.RC1	CloudVC V600R19C10	新增支持设置录播模式

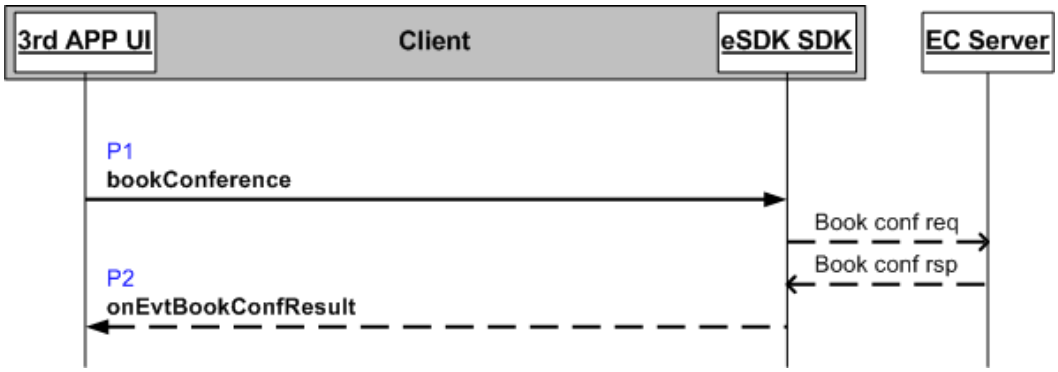
前提条件

1. 鉴权登录成功。
2. 会议环境参数已设置。

eSDK CloudVC 20.1.RC1 开发指南(Android)

流程说明

图 5-17 预约会议流程



步骤1 UI调用TsdkConferenceManager对象的bookConference()方法预约会议，参数类型为TsdkBookConfInfo类。

说明

在创建TsdkBookConfInfo对象后，需要从该对象调用方法来设置必填参数和可选参数。

1. 在预约会议时，会议方数（size）、会议类型（conf_type）、媒体类型（conf_media_type）和与会者信息（attendee_list&attendee_num）必选，其他参数可选；
2. 按具体需求填写会议方数，当实际与会者数目多于设置的方数时，服务会自动扩大会议方数，当填写方数小于3时，服务器默认会议方数为3。
3. 预约会议，会议类型应选TsdkConfType.TSDK_E_CONF_RESERVED。
4. 服务器默认时间为UTC时间，在预约时需将预约时间转换为UTC时间。

代码示例：

```
//Java code
public int bookConference(BookConferenceInfo bookConferenceInfo)
{
    Log.i(TAG, "bookConference.");

    if (bookConferenceInfo == null)
    {
        Log.e(TAG, "booKConferenceInfo obj is null");
        return -1;
    }

    TsdkBookConfInfo bookConfInfo = new TsdkBookConfInfo();

    if(bookConferenceInfo.isInstantConference())
    {
        bookConfInfo.setConfType(TsdkConfType.TSDK_E_CONF_INSTANT);
        bookConfInfo.setIsAutoProlong(1);
    }
    else
    {
        bookConfInfo.setConfType(TsdkConfType.TSDK_E_CONF_RESERVED);
    }

    bookConfInfo.setSubject(bookConferenceInfo.getSubject());
    bookConfInfo.setConfMediaType(bookConferenceInfo.getMediaType());
    bookConfInfo.setStartTime(bookConferenceInfo.getStartTime());
    bookConfInfo.setDuration(bookConferenceInfo.getDuration());
    bookConfInfo.setSize(bookConferenceInfo.getSize());
    bookConfInfo.setIsAutoRecord(bookConferenceInfo.getIs_auto()? 1:0);
    bookConfInfo.setRecordMode(bookConferenceInfo.getRecordType());
```

```
    List<TsdkAttendeeBaseInfo> attendeeList =
ConfConvertUtil.memberListToAttendeeList(bookConferenceInfo.getMemberList());
    bookConfInfo.setAttendeeList(attendeeList);
    bookConfInfo.setAttendeeNum(attendeeList.size());

    //The other parameters are optional, using the default value
    bookConfInfo.setLanguage(TsdkConfLanguage.TSDK_E_CONF_LANGUAGE_EN_US);

    int result = TsdkManager.getInstance().getConferenceManager().bookConference(bookConfInfo);
    if (result != 0)
    {
        Log.e(TAG, "bookReservedConf result ->" + result);
        return  result;
    }

    return 0;
}
```

步骤2 SDK在收到服务器返回的会议预约结果响应后，通过TsdkNotify对象中的onEvtBookConfResult()方法向UI上报预约会议结果，对应的结果参数为预约会议结果和会议信息。

说明

如果会议成功预约，其他用户可以通过“**查询会议列表**”或其他第三方方式获取该会议的信息。

代码示例：

```
//Java code
public void handleBookConfResult(TsdkCommonResult result, TsdkConfBaseInfo confBaseInfo){
    Log.i(TAG, "onBookReservedConfResult");
    if ((result == null) || (confBaseInfo == null))
    {
        Log.e(TAG, "book conference is failed, unknown error.");
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.BOOK_CONF_FAILED, -1);
        return;
    }
    if (result.getResult() != TupConfParam.CONF_RESULT.TUP_SUCCESS)
    {
        Log.e(TAG, "book conference is failed, return ->" + result.getResult());
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.BOOK_CONF_FAILED,
result.getResult());
        return;
    }
    Log.i(TAG, "book conference is success.");
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.BOOK_CONF_SUCCESS,
result.getResult());
}
```

----结束

注意事项

无。

5.5.2.2 创建即时会议

应用场景

移动用户创建立即会议。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

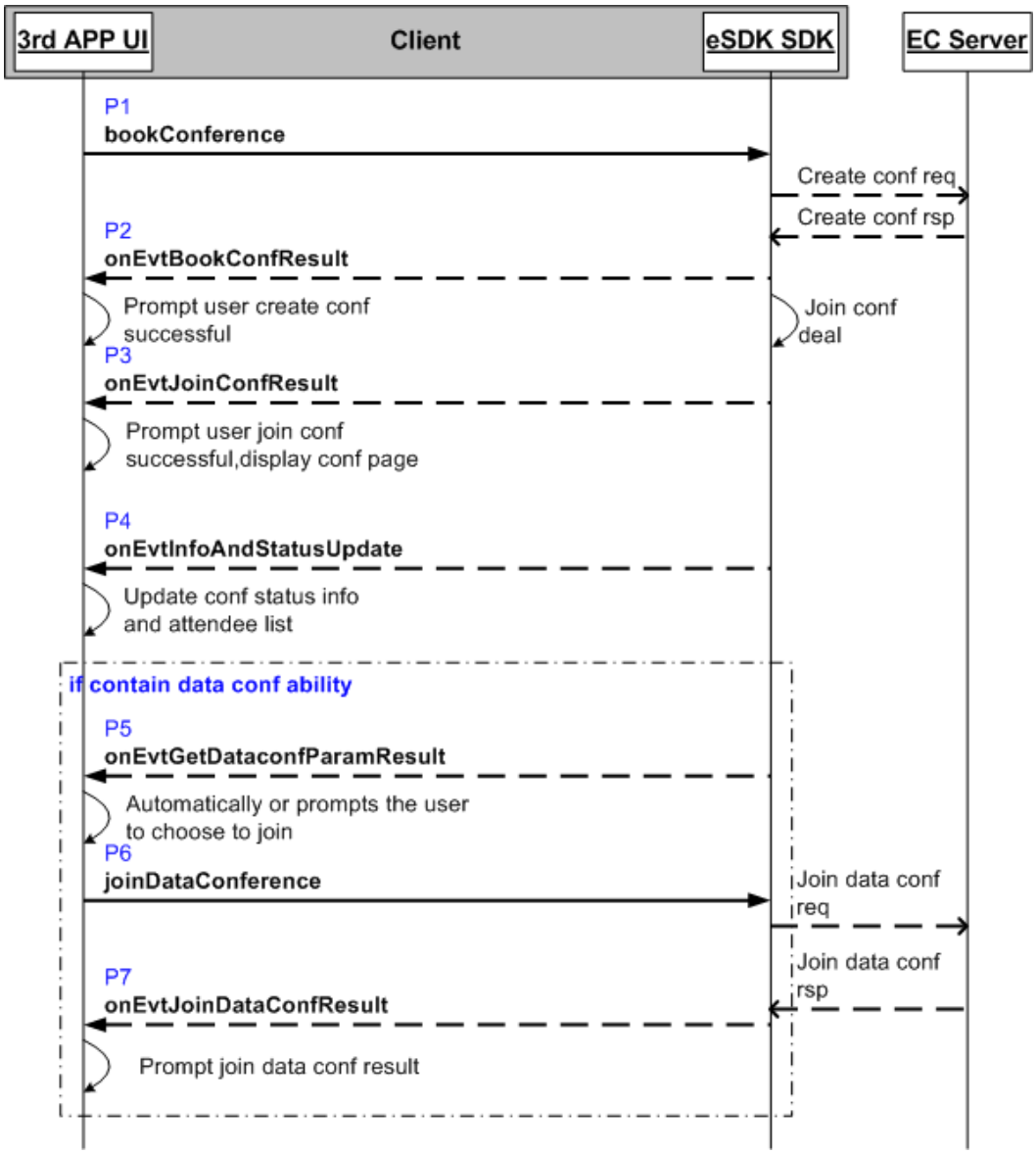
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	eSDK VC 19.1.RC1	CloudVC V600R19C10	新增支持设置录播模式

前提条件

1. 鉴权登录成功；
2. SIP号码已成功注册；
3. 会议环境参数已设置。

流程说明

图 5-18 创建立即会议流程



步骤1 UI调用TsdkConferenceManager对象中的bookConference()方法创建立即会议。参数类型为TsdkBookConfInfo类。

说明

在创建TsdkBookConfInfo对象后，需要从该对象调用方法来设置必填参数和可选参数。

1. 在预约会议时，会议方数（size）、会议类型（conf_type）、媒体类型（conf_media_type）和与会者信息（attendee_list&attendee_num）必选，其他参数可选；
2. 按具体需求填写会议方数，当实际与会者数目多于设置的方数时，服务会自动扩大会议方数，当填写方数小于3时，服务器默认会议方数为3。
3. 预约会议，会议类型应选TsdkConfType.TSDK_E_CONF_INSTANT。
4. 服务器默认时间为UTC时间，在预约时需将预约时间转换为UTC时间。

代码示例：

```
//Java code
public int bookConference(BookConferenceInfo bookConferenceInfo)
{
    Log.i(TAG, "bookConference.");

    if (bookConferenceInfo == null)
    {
        Log.e(TAG, "booKConferenceInfo obj is null");
        return -1;
    }

    TsdkBookConfInfo bookConfInfo = new TsdkBookConfInfo();

    if(bookConferenceInfo.isInstantConference())
    {
        bookConfInfo.setConfType(TsdkConfType.TSDK_E_CONF_INSTANT);
        bookConfInfo.setIsAutoProlong(1);
    }
    else
    {
        bookConfInfo.setConfType(TsdkConfType.TSDK_E_CONF_RESERVED);
    }

    bookConfInfo.setSubject(bookConferenceInfo.getSubject());
    bookConfInfo.setConfMediaType(bookConferenceInfo.getMediaType());
    bookConfInfo.setStartTime(bookConferenceInfo.getStartTime());
    bookConfInfo.setDuration(bookConferenceInfo.getDuration());
    bookConfInfo.setSize(bookConferenceInfo.getSize());
    bookConfInfo.setIsAutoRecord(bookConferenceInfo.getIs_auto()? 1:0);
    bookConfInfo.setRecordMode(bookConferenceInfo.getRecordType());

    List<TsdkAttendeeBaseInfo> attendeeList =
ConfConvertUtil.memberListToAttendeeList(bookConferenceInfo.getMemberList());
    bookConfInfo.setAttendeeList(attendeeList);
    bookConfInfo.setAttendeeNum(attendeeList.size());

    //The other parameters are optional, using the default value
    bookConfInfo.setLanguage(TsdkConfLanguage.TSDK_E_CONF_LANGUAGE_EN_US);

    int result = TsdkManager.getInstance().getConferenceManager().bookConference(bookConfInfo);
    if (result != 0)
    {
        Log.e(TAG, "bookReservedConf result ->" + result);
        return  result;
    }

    return 0;
}
```

步骤2 SDK在收到服务器返回的立即会议创建成功后，通过TsdkNotify对象中的onEvtBookConfResult()方法向UI上报会议创建结果通知。

代码示例：

```
//Java code
public void handleBookConfResult(TsdkCommonResult result, TsdkConfBaseInfo confBaseInfo){
    Log.i(TAG, "onBookReservedConfResult");
    if ((result == null) || (confBaseInfo == null))
    {
        Log.e(TAG, "book conference is failed, unknown error.");
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.BOOK_CONF_FAILED, -1);
        return;
    }
    if (result.getResult() != TupConfParam.CONF_RESULT.TUP_SUCCESS)
    {
        Log.e(TAG, "book conference is failed, return ->" + result.getResult());
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.BOOK_CONF_FAILED,
result.getResult());
        return;
```

```
    }
    Log.i(TAG, "book conference is success.");
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.BOOK_CONF_SUCCESS,
result.getResult());
}
```

步骤3 SDK在收到服务器返回的加入会议响应后，通过TsdkNotify对象中的onEvtJoinConfResult()方法向UI上报会议加入结果事件。并返回会议对象TsdkConference，后续会控时使用。此时UI可跳转至会议界面。

说明

在加入会议时，会请求会议权限，若请求失败，则通过TsdkNotify对象中的onEvtRequestConfRightFailed()方法向UI上报会议权限请求失败通知，UI提示申请会控权限失败，若请求成功，则不上报。

代码示例：

```
//Java code
public void handleJoinConfResult(TsdkConference tsdkConference, TsdkCommonResult commonResult,
TsdkJoinConfIndInfo tsdkJoinConfIndInfo
    Log.i(TAG, "handleJoinConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    if (result == 0)
    {
        this.currentConference = tsdkConference;
        this.memberList = null;
        this.self = null;
        TsdkCall tsdkCall = tsdkConference.getCall();
        if (null != tsdkCall) {
            Session newSession =
CallMgr.getInstance().getCallSessionByCallID(tsdkCall.getCallInfo().getCallId());
            if (null == newSession) {
                newSession = new Session(tsdkCall);
                CallMgr.getInstance().putCallSessionToMap(newSession);
            }
            if (tsdkCall.getCallInfo().getIsVideoCall() == 1) {
                VideoMgr.getInstance().initVideoWindow(tsdkCall.getCallInfo().getCallId());
            }
        }
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_CONF_SUCCESS,
tsdkConference.getHandle() + "");
    }
    else
    {
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_CONF_FAILED, result);
    }
}
```

步骤4 SDK收到会议状态刷新通知，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及会议状态刷新事件，UI刷新会议状态和成员列表。

说明

详细流程参见“**会议信息及会议状态更新**”描述。

代码示例：

步骤5 若会议包含数据会议能力，SDK通过TsdkNotify对象中的onEvtGetDataconfParamResult()方法向UI上报获取数据会议参数结果。

代码示例：

```
//Java code
public void  handleGetDataConfParamsResult(TsdkConference tsdkConference, TsdkCommonResult
commonResult){
```



```
    Log.i(TAG, "handleJoinConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.GET_DATA_CONF_PARAM_RESULT,
result);
}
```

步骤6 此时UI可选择自动加入或用户选择加入数据会议，调用TsdkConference对象中的joinDataConference()方法加入数据会议。

代码示例：

```
//Java code
public int joinDataConf()
{
    if (null == currentConference)
    {
        Log.e(TAG,  "join data conf failed, currentConference is null ");
        return -1;
    }

    int result =  currentConference.joinDataConference();

    return result;
}
```

步骤7 SDK在收服务器加入数据会议响应后，通过TsdkNotify对象中的onEvtJoinDataConfResult()方法向UI上报数据会议加入结果事件，若成功，则UI刷新界面，提示加入数据会议成功，若失败，则提示加入数据会议失败。

说明

在加入数据会议后，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及状态更新事件，UI刷新会议状态和成员列表。

代码示例：

```
//Java code
public void  handleJoinDataConfResult(TsdkConference tsdkConference, TsdkCommonResult commonResult)
{
    Log.i(TAG, "handleJoinDataConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_DATA_CONF_RESULT, result);
}
```

----结束

注意事项

无。

5.5.2.3 查询会议列表

应用场景

用户查询自己“创建”的和“待参加”的预约会议信息。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

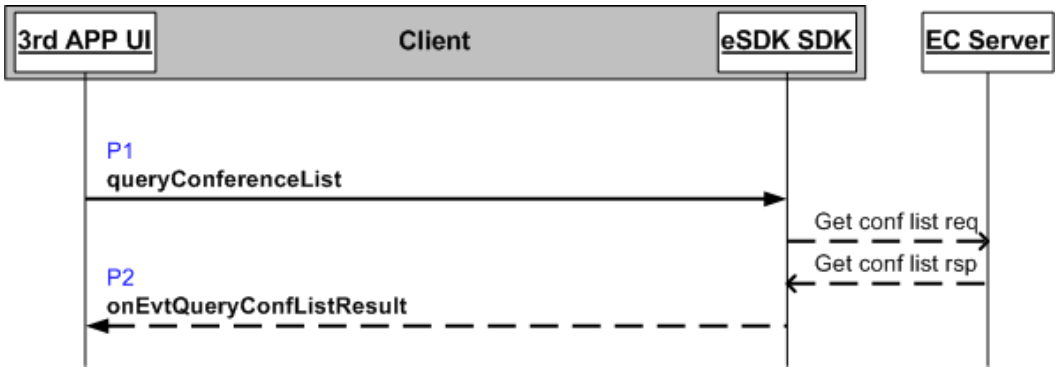
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

1. 鉴权登录成功；
2. 会议环境参数已设置。

流程说明

图 5-19 查询会议列表流程



步骤1 UI调用TsdkConferenceManager对象的queryConferenceList()查询会议列表，参数为获取会议列表信息，参数类型为TsdkQueryConfListReq类。

说明

在创建TsdkQueryConfListReq对象后，需要从该对象调用方法来设置参数。

1. 会议权限(conf_right)，使用setConfRight()方法设置，参考枚举类ConfctrlConfRight，用于指定要查询的会议权限类型，包含查询创建的会议、待参加的会议或创建和待参加的会议，可选填；
2. 指定返回的页面索引(page_index)，使用setPageIndex()方法设置，参数类型为int，取值从1开始，建议与应用程序与会议列表的页签对应，必须要有明确值；
3. 指定每页返回的记录数(page_size)，使用setPageSize()方法设置，参数类型为int，建议与应用程序会议列表个数相同，必须要有明确值。
4. 是否包含已结束的会议(is_include_end)，使用setIsIncludeEnd()方法设置，参数类型为int。
5. 制定查询截止时间queryEndTime，使用setQueryEndTime()方法设置，参数类型为String，如果设置就查询到这个时间点之前的会议列表，如果不设置就查询到所有的会议列表。
6. 返回来的时间为UTC时间，UI进行页面呈现之前需进行时间处理。

代码示例：

```
//Java code
public int queryMyConfList(ConfConstant.ConfRight myRight)
{
    Log.i(TAG, "query my conf list.");
    TsdkConfRight tupConfRight;
    switch (myRight)
    {
        case MY_CREATE:
            tupConfRight = TsdkConfRight.TSDK_E_CONF_RIGHT_CREATE;
            break;
        case MY_JOIN:
            tupConfRight = TsdkConfRight.TSDK_E_CONF_RIGHT_JOIN;
            break;
        case MY_CREATE_AND_JOIN:
            tupConfRight = TsdkConfRight.TSDK_E_CONF_RIGHT_CREATE_JOIN;
            break;
        default:
            tupConfRight = TsdkConfRight.TSDK_E_CONF_RIGHT_CREATE_JOIN;
            break;
    }
    TsdkQueryConfListReq queryReq = new TsdkQueryConfListReq();
    queryReq.setPageSize(ConfConstant.PAGE_SIZE);
    queryReq.setPageIndex(1);
    queryReq.setIsIncludeEnd(0);
    queryReq.setConfRight(tupConfRight);
    int result = TsdkManager.getInstance().getConferenceManager().queryConferenceList(queryReq);
    if (result != 0)
    {
        Log.e(TAG, "getConfList result ->" + result);
        return  result;
    }
    return 0;
}
```

步骤2 SDK在收到服务器返回的查询会议列表响应后，通过TsdkNotify对象中的onEvtQueryConfListResult()方法向UI上报查询会议列表结果，参数包括会议列表信息和操作结果。

说明

1、查询会议列表只返回会议的概要信息，如需查询会议详情（包括与会者信息），需要“**查询会议详情**”；

2、会议的主席密码需要“**查询会议详情**”获取。

代码示例：

```
//Java code
public void handleQueryConfListResult(TsdkCommonResult result, TsdkConfListInfo confList){
```



```
    Log.i(TAG, "onGetConfListResult");
    if (result == null)
    {
        Log.e(TAG, "get conference list is failed, unknown error.");
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.QUERY_CONF_LIST_FAILED, -1);
        return;
    }
    else if (result.getResult() != 0)
    {
        Log.e(TAG, "get conference list is failed, return ->" + result.getReasonDescription());
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.QUERY_CONF_LIST_FAILED,
result.getResult());
        return;
    }
    List<ConfBaseInfo> confBaseInfoList = new ArrayList<>();
    List<TsdkConfBaseInfo> tsdkConfBaseInfos = confList.getConfInfoList();
    if (null != tsdkConfBaseInfos) {
        for (TsdkConfBaseInfo confInfo : tsdkConfBaseInfos) {
            ConfBaseInfo confBaseInfo = new ConfBaseInfo();
            confBaseInfo.setSize(confInfo.getSize());
            confBaseInfo.setConfID(confInfo.getConfId());
            confBaseInfo.setSubject(confInfo.getSubject());
            confBaseInfo.setAccessNumber(confInfo.getAccessNumber());
            confBaseInfo.setChairmanPwd(confInfo.getChairmanPwd());
            confBaseInfo.setGuestPwd(confInfo.getGuestPwd());
            confBaseInfo.setSchedulerNumber(confInfo.getScheduserAccount());
            confBaseInfo.setSchedulerName(confInfo.getScheduserName());
            confBaseInfo.setStartTime(confInfo.getStartTime());
            confBaseInfo.setEndTime(confInfo.getEndTime());
            confBaseInfo.setMediaType(ConfConvertUtil.convertConfMediaType(confInfo.getConfMediaType()));
            confBaseInfo.setConfState(ConfConvertUtil.convertConfctrlConfState(confInfo.getConfState()));
            confBaseInfoList.add(confBaseInfo);
        }
    }
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.QUERY_CONF_LIST_SUCCESS,
confBaseInfoList);
}
```

----结束

注意事项

无。

5.5.2.4 查询会议详情

应用场景

用户查询指定会议的详细信息。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

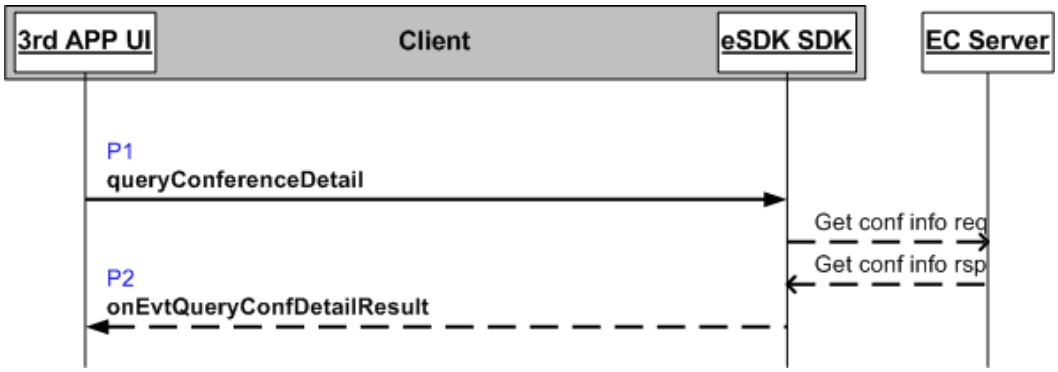
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

1. 鉴权登录成功；
2. 会议环境参数已设置。

流程说明

图 5-20 查询会议详情流程



步骤1 UI调用TsdkConferenceManager对象的queryConferenceDetail()方法查询会议详情，参数为获取会议详细信息，参数类型为TsdkQueryConfDetailReq类。

说明

在创建TsdkQueryConfDetailReq对象后，需要从该对象调用方法来设置参数。

- 请求与会者列表页索引，通过setPageIndex()方法设置，取值从1开始，建议与应用程序与会者列表的页签对应；
- 需要查询的会议ID，通过查询列表或其他方式获取；
- 与会者列表每页的与会者个数，通过setPageSize()方法设置，建议与应用程序与会者列表个数相同。

代码示例：

```
//Java code
public int queryConfDetail(String confID)
{
    Log.i(TAG,  "query conf detail");
    TsdkQueryConfDetailReq queryReq = new TsdkQueryConfDetailReq();
    queryReq.setConfId(confID);
    int result = TsdkManager.getInstance().getConferenceManager().queryConferenceDetail(queryReq);
    if (result != 0)
    {
        Log.e(TAG, "getConfInfo result ->" + result);
        return  result;
    }
    return result;
}
```

步骤2 SDK在收到服务器返回的查询会议详情响应后，通过TsdkNotify对象中的onEvtQueryConfDetailResult()方法向UI上报查询结果，参数包括会议详情信息和操作结果。

代码示例：

```
//Java code
public void handleQueryConfDetailResult(TsdkCommonResult result, TsdkConfDetailInfo tsdkConfDetailInfo)
{
    Log.i(TAG, "onGetConfInfoResult");
    if ((result == null) || (tsdkConfDetailInfo == null))
    {
        Log.e(TAG, "get conference detail is failed, unknown error.");
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.QUERY_CONF_DETAIL_FAILED, -1);
        return;
    }
    if (result.getResult() != TupConfParam.CONF_RESULT.TUP_SUCCESS)
    {
        Log.e(TAG, "get conference detail is failed, return ->" + result.getResult());
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.QUERY_CONF_DETAIL_FAILED,
result.getResult());
        return;
    }
    ConfDetailInfo confDetailInfo = new ConfDetailInfo();
    confDetailInfo.setSize(tsdkConfDetailInfo.getConfInfo().getSize());
    confDetailInfo.setConfID(tsdkConfDetailInfo.getConfInfo().getConfId());
    confDetailInfo.setSubject(tsdkConfDetailInfo.getConfInfo().getSubject());
    confDetailInfo.setAccessNumber(tsdkConfDetailInfo.getConfInfo().getAccessNumber());
    confDetailInfo.setChairmanPwd(tsdkConfDetailInfo.getConfInfo().getChairmanPwd());
    confDetailInfo.setGuestPwd(tsdkConfDetailInfo.getConfInfo().getGuestPwd());
    confDetailInfo.setSchedulerNumber(tsdkConfDetailInfo.getConfInfo().getScheduserAccount());
    confDetailInfo.setSchedulerName(tsdkConfDetailInfo.getConfInfo().getScheduserName());
    confDetailInfo.setStartTime(tsdkConfDetailInfo.getConfInfo().getStartTime());
    confDetailInfo.setEndTime(tsdkConfDetailInfo.getConfInfo().getEndTime());

confDetailInfo.setMediaType(ConfConvertUtil.convertConfMediaType(tsdkConfDetailInfo.getConfInfo().getCo
nfMediaType()));

confDetailInfo.setConfState(ConfConvertUtil.convertConfctrlConfState(tsdkConfDetailInfo.getConfInfo().getC
onfState()));
    List<Member> memberList =
ConfConvertUtil.convertAttendeeInfoList(tsdkConfDetailInfo.getAttendeeList());
    confDetailInfo.setMemberList(memberList);
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.QUERY_CONF_DETAIL_SUCCESS,
confDetailInfo);
}
```

----结束

注意事项

无。

5.5.3 会议接入

5.5.3.1 会议列表一键入会

应用场景

用户在移动客户端通过会议列表一键入会的方式加入会议。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

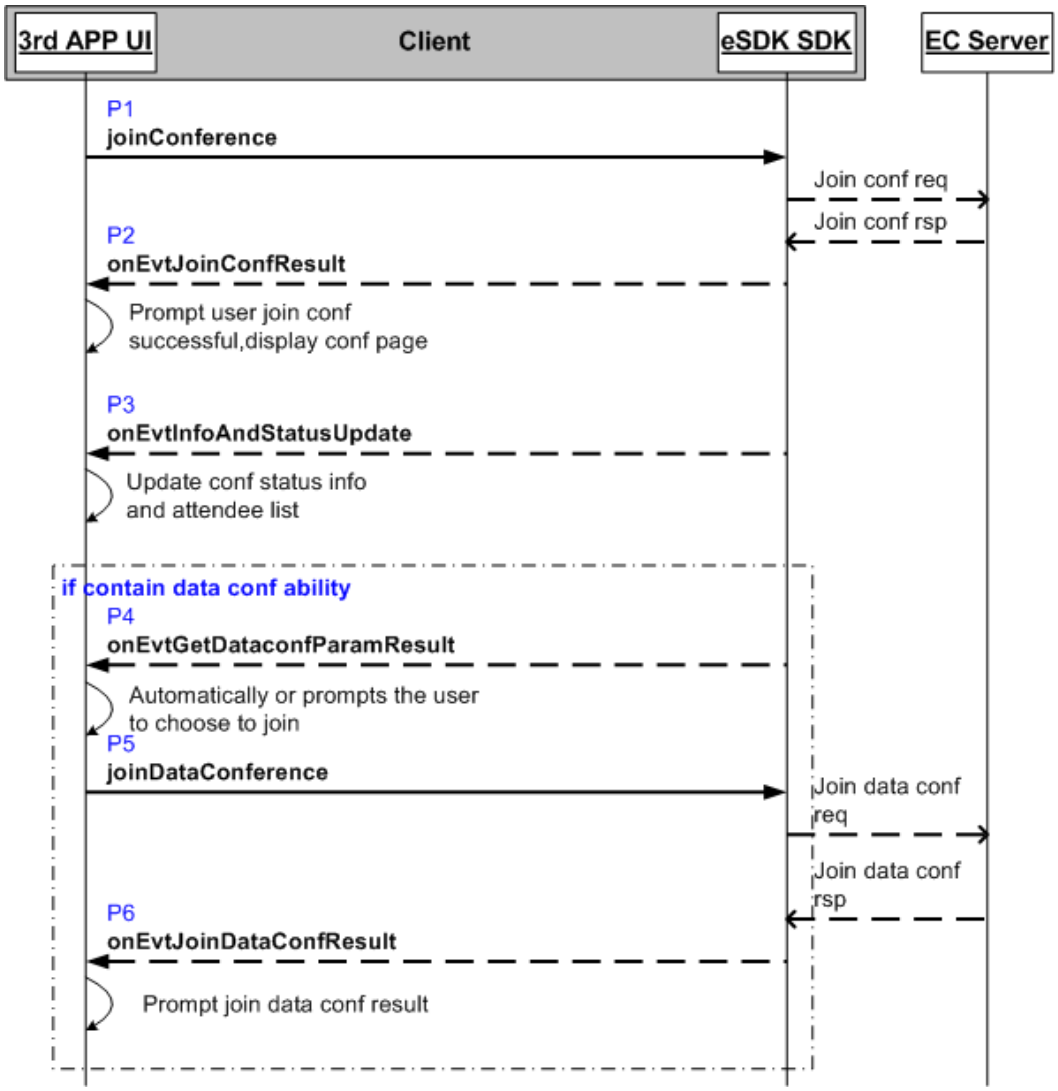
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

1. 鉴权登录成功；
2. SIP号码已成功注册；
3. 会议环境参数已设置。

流程说明

图 5-21 会议列表一键入会流程



步骤1 UI调用TsdkConferenceManager对象中的joinConference()方法主动加入会议。SDK发送加入会议请求至服务器。

说明

在加入会议时，需要传入的参数为：是否接入视频会议以及入会参数：

1. 入会参数中conf_password、conf_id以及access_number必选；
2. 入会号码（joinNumber）如果不填，则使用自己软终端号码入会；
3. 会议对应的呼叫ID（接口会同步返回呼叫ID,需要记录下），在使用SIP终端号码入会时有效。

代码示例：

```
//Java code
public int joinConf(TsdkConfJoinParam confJoinParam, boolean isVideo, String joinNumber)
{
    Log.i(TAG,  "join conf.");
    int result = TsdkManager.getInstance().getConferenceManager().joinConference(confJoinParam, isVideo,
```

```
joinNumber);
    if (result != 0)
    {
        Log.e(TAG, "joinConf result ->" + result);
        currentConference = null;
        return result;
    }
    return 0;
}
```

步骤2 SDK在收到服务器返回的加入会议响应后，通过TsdkNotify对象中的onEvtJoinConfResult()方法向UI上报会议加入结果事件。并返回会议对象TsdkConference，后续会控时使用，此时， UI可跳转至会议界面。

说明

在加入会议时，会请求会议权限，若请求失败，则通过TsdkNotify对象中的onEvtRequestConfRightFailed()方法向UI上报会议权限请求失败通知，UI提示申请会控权限失败，若请求成功，则不上报。

代码示例：

```
//Java code
public void handleJoinConfResult(TsdkConference tsdkConference, TsdkCommonResult commonResult,
TsdkJoinConfIndInfo tsdkJoinConfIndInfo
    Log.i(TAG, "handleJoinConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    if (result == 0)
    {
        this.currentConference = tsdkConference;
        this.memberList = null;
        this.self = null;
        TsdkCall tsdkCall = tsdkConference.getCall();
        if (null != tsdkCall) {
            Session newSession =
CallMgr.getInstance().getCallSessionByCallID(tsdkCall.getCallInfo().getCallId());
            if (null == newSession) {
                newSession = new Session(tsdkCall);
                CallMgr.getInstance().putCallSessionToMap(newSession);
            }
            if (tsdkCall.getCallInfo().getIsVideoCall() == 1) {
                VideoMgr.getInstance().initVideoWindow(tsdkCall.getCallInfo().getCallId());
            }
        }
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_CONF_SUCCESS,
tsdkConference.getHandle() + "");
    }
    else
    {
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_CONF_FAILED, result);
    }
}
```

步骤3 SDK收到会议状态刷新通知，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及状态更新事件，UI刷新会议状态和成员列表。

说明

详细流程参见“**会议信息及会议状态更新**”描述。

步骤4 若会议包含数据会议能力，SDK通过TsdkNotify对象中的onEvtGetDataconfParamResult()方法向UI上报获取数据会议参数结果。

代码示例：

```
//Java code
public void  handleGetDataConfParamsResult(TsdkConference tsdkConference, TsdkCommonResult
```



```
commonResult){
    Log.i(TAG, "handleJoinConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.GET_DATA_CONF_PARAM_RESULT,
result);
}
```

步骤5 此时UI可选择自动加入或用户选择加入数据会议，调用TsdkConference对象中的joinDataConference()方法加入数据会议。

代码示例：

```
//Java code
public int joinDataConf()
{
    if (null == currentConference)
    {
        Log.e(TAG,  "join data conf failed, currentConference is null ");
        return -1;
    }

    int result =  currentConference.joinDataConference();

    return result;
}
```

步骤6 SDK在收服务器加入数据会议响应后，通过TsdkNotify对象中的onEvtJoinDataConfResult()方法向UI上报数据会议加入结果事件，若成功，则UI刷新界面，提示加入数据会议成功，若失败，则提示加入数据会议失败。

说明

在加入数据会议后，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及状态更新事件，UI刷新会议状态和成员列表。

代码示例：

```
//Java code
public void  handleJoinDataConfResult(TsdkConference tsdkConference, TsdkCommonResult commonResult)
{
    Log.i(TAG, "handleJoinDataConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_DATA_CONF_RESULT, result);
}
```

----结束

注意事项

无。

5.5.3.2 会议接入码入会

应用场景

用户由第三方途径获取会议信息，通过输入会议号和接入码的方式加入会议。

说明

1. 因不支持从拨号盘拨打“会议号”+“接入码”的方式直接加入会议，所以应用程序界面需要提供“接入会议”的单独入口；
2. 与“**会议列表一键入会**”的接口调用流程相同，不同在于用户界面入口。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

参见“**会议列表一键入会**”。

流程说明

参见“**会议列表一键入会**”。

注意事项

无。

5.5.3.3 统一会议接入号入会

应用场景

用户由第三方途径获取会议信息，通过拨打统一会议接入号，使用IVR导航的方式加入会议。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

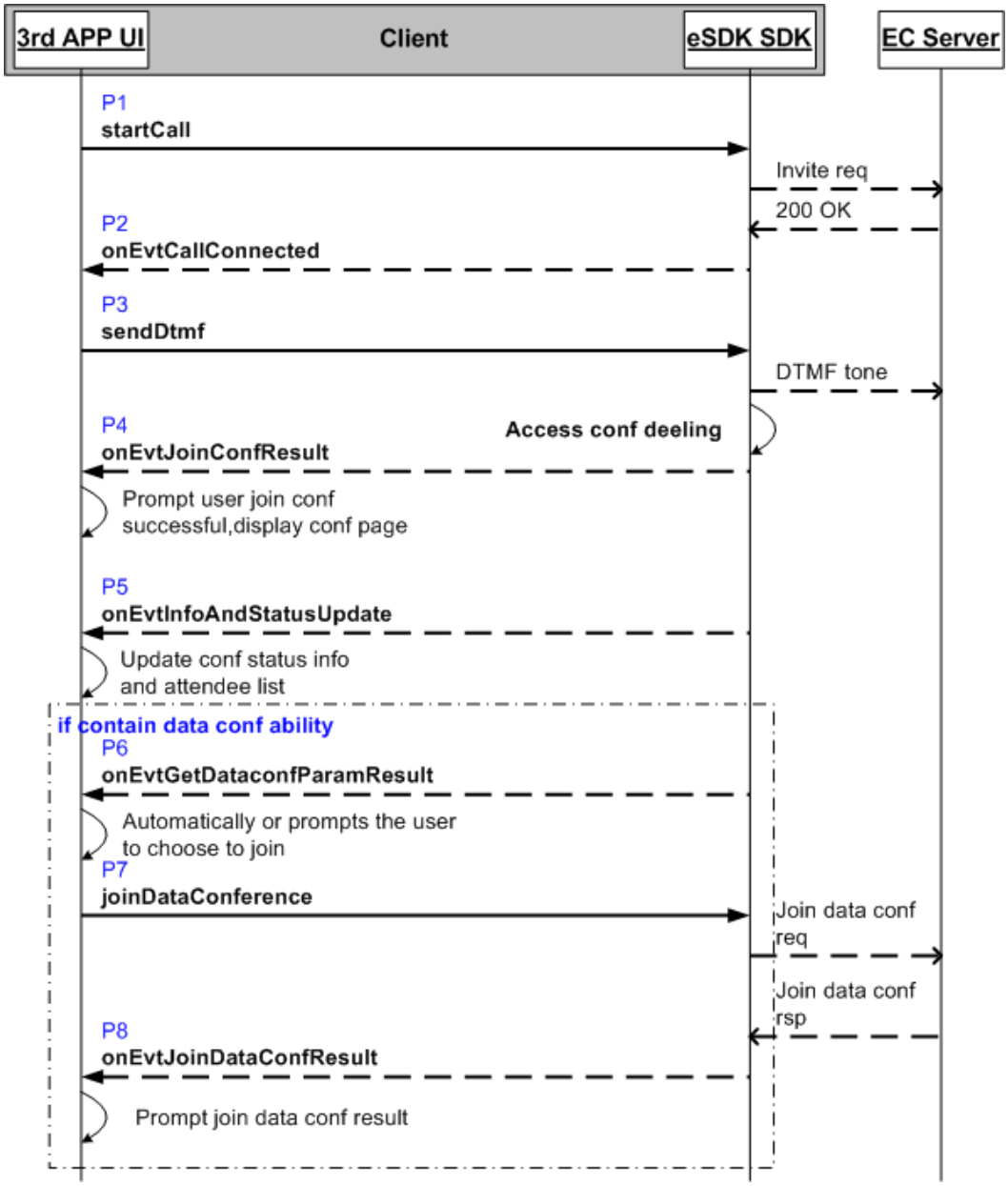
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

1. 鉴权登录成功；
2. SIP号码已成功注册；
3. 会议环境参数已设置。

流程说明

图 5-22 统一会议接入号入会流程



步骤1 UI调用TsdkCallManager对象中的startCall()方法来发起一个呼叫请求，被叫号码为“统一会议接入号”。

代码示例：

```
// Java code
TsdkCall call = TsdkManager.getInstance().getCallManager().startCall(toNumber, isVideoCall);
```

步骤2 若呼叫建立成功，SDK通过TsdkNotify对象中的onEvtCallConnected()方法向UI上报呼叫建立事件，与普通音视频呼叫相同。

代码示例：

```
//Java code
public void onEvtCallConnected(TsdkCall call){
```



```
    Log.i(TAG, "onCallConnected");
    if (null == call)
    {
        Log.e(TAG, "call obj is null");
        return;
    }

    CallInfo callInfo = getCallInfo(call);
    Session callSession = getCallSessionByCallID(call.getCallInfo().getCallId());
    if (callSession == null)
    {
        Log.e(TAG, "call session obj is null");
        return;
    }

    if (callInfo.isVideoCall())
    {
        callSession.setCallStatus(CallConstant.CallStatus.VIDEO_CALLING);
    }
    else
    {
        callSession.setCallStatus(CallConstant.CallStatus.AUDIO_CALLING);
    }

    mCallNotification.onCallEventNotify(CallConstant.CallEvent.CALL_CONNECTED, callInfo);
}
```

步骤3 用户根据服务器的语言提示，在二次拨号界面输入会议接入码和密码（密码必须以"#"号结尾），UI调用TsdkCall对象中的sendDtmf()方法在通话中发送DTMF信号完成会议接入码和密码发送，SDK完成入会交互处理。

代码示例：

```
//Java code
public boolean reDial(int code)
{
    TsdkDtmfTone tsdkDtmfTone = TsdkDtmfTone.enumOf(code);
    LogUtil.d(TAG, "Dtmf Tone ： " + tsdkDtmfTone.getIndex());
    int result = tsdkCall.sendDtmf(tsdkDtmfTone);
    if (result != 0)
    {
        LogUtil.e(TAG, "sendDTMF return failed, result = " + result);
        return false;
    }
    return true;
}
```

步骤4 SDK在收到服务器返回的加入会议响应后，通过TsdkNotify对象中的onEvtJoinConfResult()方法向UI上报会议加入结果事件。并返回会议对象TsdkConference，后续会控时使用，此时， UI可跳转至会议界面。

说明

在加入会议时，会请求会议权限，若请求失败，则通过TsdkNotify对象中的onEvtRequestConfRightFailed()方法向UI上报会议权限请求失败通知，UI提示申请会控权限失败，若请求成功，则不上报。

代码示例：

```
//Java code
public void handleJoinConfResult(TsdkConference tsdkConference, TsdkCommonResult commonResult,
TsdkJoinConfIndInfo tsdkJoinConfIndInfo
    Log.i(TAG, "handleJoinConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    if (result == 0)
    {
        this.currentConference = tsdkConference;
```

```
        this.memberList = null;
        this.self = null;
        TsdkCall tsdkCall = tsdkConference.getCall();
        if (null != tsdkCall) {
            Session newSession =
CallMgr.getInstance().getCallSessionByCallID(tsdkCall.getCallInfo().getCallId());
            if (null == newSession) {
                newSession = new Session(tsdkCall);
                CallMgr.getInstance().putCallSessionToMap(newSession);
            }
            if (tsdkCall.getCallInfo().getIsVideoCall() == 1) {
                VideoMgr.getInstance().initVideoWindow(tsdkCall.getCallInfo().getCallId());
            }
        }
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_CONF_SUCCESS,
tsdkConference.getHandle() + "");
    }
    else
    {
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_CONF_FAILED, result);
    }
}
```

步骤5 SDK收到会议状态刷新通知，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及状态更新事件，UI刷新会议状态和成员列表。

说明

详细流程参见“**会议信息及会议状态更新**”描述。

步骤6 若会议包含数据会议能力，SDK通过TsdkNotify对象中的onEvtGetDataconfParamResult()方法向UI上报获取数据会议参数结果。

代码示例：

```
//Java code
public void  handleGetDataConfParamsResult(TsdkConference tsdkConference, TsdkCommonResult
commonResult){
    Log.i(TAG, "handleJoinConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.GET_DATA_CONF_PARAM_RESULT,
result);
}
```

步骤7 此时UI可选择自动加入或用户选择加入数据会议，调用TsdkConference对象中的joinDataConference()方法加入数据会议。

代码示例：

```
//Java code
public int joinDataConf()
{
    if (null == currentConference)
    {
        Log.e(TAG,  "join data conf failed, currentConference is null ");
        return -1;
    }

    int result =  currentConference.joinDataConference();

    return result;
}
```

步骤8 SDK在收服务器加入数据会议响应后，通过TsdkNotify对象中的onEvtJoinDataConfResult()方法向UI上报数据会议加入结果事件，若成功，则UI刷新界面，提示加入数据会议成功，若失败，则提示加入数据会议失败。

说明

在加入数据会议后，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及状态更新事件，UI刷新会议状态和成员列表。

代码示例：

```
//Java code
public void  handleJoinDataConfResult(TsdkConference tsdkConference, TsdkCommonResult commonResult)
{
    Log.i(TAG, "handleJoinDataConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_DATA_CONF_RESULT, result);
}
```

----结束

注意事项

无。

5.5.3.4 被邀接入会议

应用场景

会议主席邀请新的与会者加入会议。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

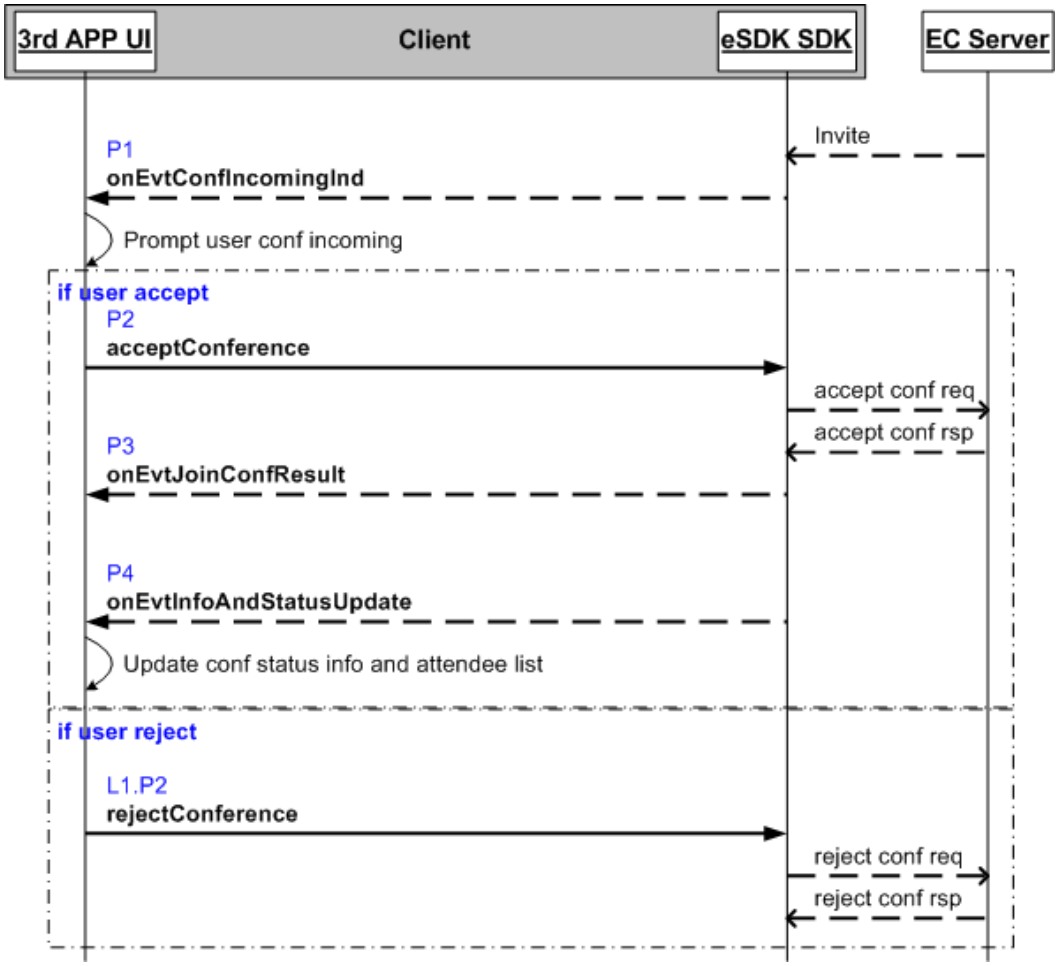
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

与会者通过主席身份加入会议，或已申请获取为主席。

流程说明

图 5-23 被邀接入会议流程



步骤1 被邀与会者侧SDK收到会议来电请求，通过TsdkNotify对象中的onEvtConfIncomingInd()方法向UI上报会议来电事件，UI提示用户会议来电。

代码示例：

```
//Java code
public void handleConfIncomingInd(TsdkConference conference){
    if (null == conference)
    {
        return;
    }
    currentConference = conference;
    TsdkCall tsdkCall = conference.getCall();
    if (null != tsdkCall) {
        Session newSession = CallMgr.getInstance().getCallSessionByCallID(tsdkCall.getCallInfo().getCallId());
        if (null == newSession) {
            newSession = new Session(tsdkCall);
            CallMgr.getInstance().putCallSessionToMap(newSession);
        }
    }

mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.CONF_INCOMING_TO_CALL_INCOMING,
conference);
}
```


步骤2 被邀与会者侧UI调用TsdkConference对象中的acceptConference()方法接听会议来电。

说明

若用户拒绝接听会议来电，则通过调用TsdkConference对象中的rejectConference()方法拒接会议来电。

代码示例：

```
//Java code
public int acceptConf(boolean isVideo)
{
    Log.i(TAG,  "accept conf.");

    if (null == currentConference)
    {
        Log.i(TAG,  "accept conf, currentConference is null ");
        return 0;
    }

    int result = currentConference.acceptConference(isVideo);
    if (result == 0) {
        Log.i(TAG,  "accept conf");
    }

    return result;
}
```

```
//Java code
public int rejectConf()
{
    Log.i(TAG,  "reject conf.");

    if (null == currentConference)
    {
        Log.i(TAG,  "reject conf, currentConference is null ");
        return 0;
    }

    int result = currentConference.rejectConference();
    if (result == 0) {
        currentConference = null;
    }

    return result;
}
```

步骤3 SDK在收到服务器返回的加入会议响应后，通过TsdkNotify对象中的onEvtJoinConfResult()方法向UI上报会议加入结果事件。并返回会议对象TsdkConference，后续会控时使用，此时， UI可跳转至会议界面。

说明

在加入会议时，会请求会议权限，若请求失败，则通过TsdkNotify对象中的onEvtRequestConfRightFailed()方法向UI上报会议权限请求失败通知，UI提示申请会控权限失败，若请求成功，则不上报。

代码示例：

```
//Java code
public void handleJoinConfResult(TsdkConference tsdkConference, TsdkCommonResult commonResult,
TsdkJoinConfIndInfo tsdkJoinConfIndInfo
    Log.i(TAG, "handleJoinConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    if (result == 0)
```



```
    {
        this.currentConference = tsdkConference;
        this.memberList = null;
        this.self = null;
        TsdkCall tsdkCall = tsdkConference.getCall();
        if (null != tsdkCall) {
            Session newSession =
CallMgr.getInstance().getCallSessionByCallID(tsdkCall.getCallInfo().getCallId());
            if (null == newSession) {
                newSession = new Session(tsdkCall);
                CallMgr.getInstance().putCallSessionToMap(newSession);
            }
            if (tsdkCall.getCallInfo().getIsVideoCall() == 1) {
                VideoMgr.getInstance().initVideoWindow(tsdkCall.getCallInfo().getCallId());
            }
        }
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_CONF_SUCCESS,
tsdkConference.getHandle() + "");
    }
    else
    {
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_CONF_FAILED, result);
    }
}
```

步骤4 SDK收到会议状态刷新通知，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及状态更新事件，UI刷新会议状态和成员列表。

说明

详细流程参见“**会议信息及会议状态更新**”描述。

----结束

注意事项

无。

5.5.3.5 匿名加入会议

应用场景

用户在未注册EC账号时，通过匿名方式加入一个会议。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

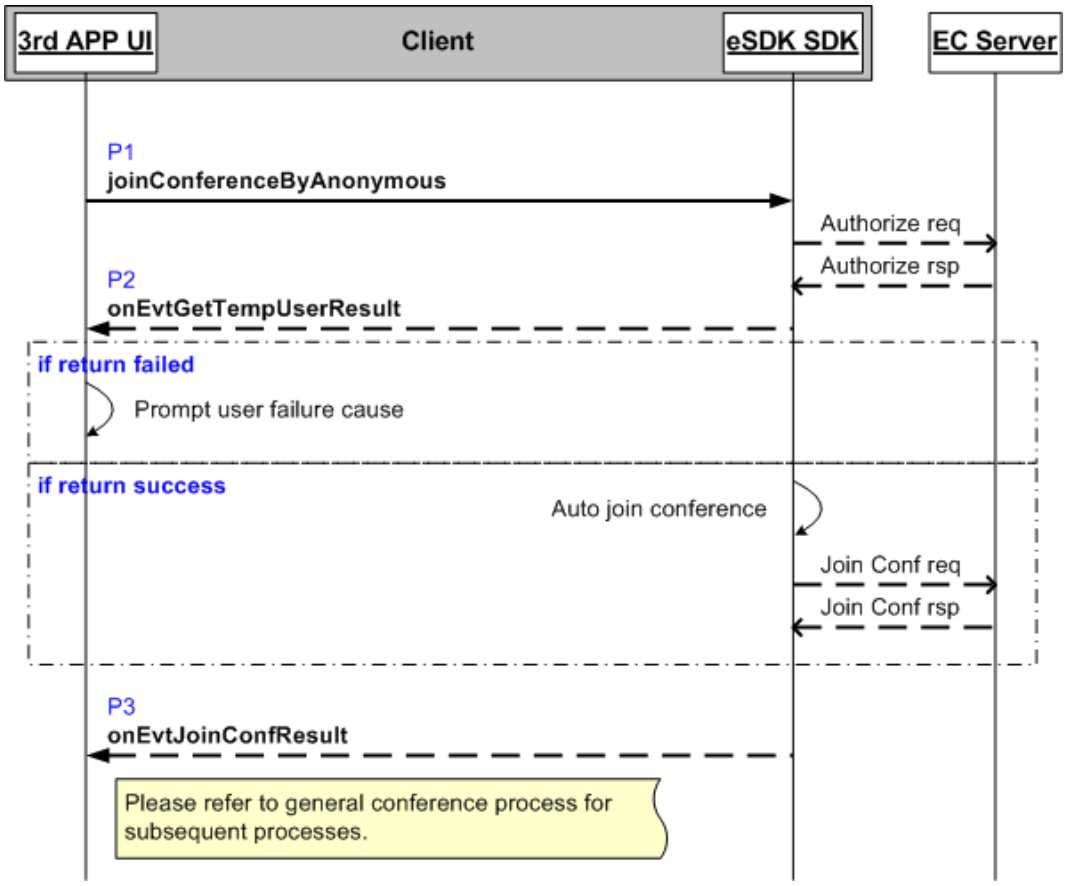
eSDK CloudVC 20.1.RC1 开发指南(Android)

前提条件

会议已经创建，且用户通过第三方方式获取到会议ID和密码。

流程说明

图 5-24 匿名加入会议流程



步骤1 UI调用TsdkConferenceManager对象中的joinConferenceByAnonymous()方法来匿名加入会议。

说明

参数TsdkConfAnonymousJoinParam 中的服务器地址和端口，指会议服务器的地址和端口。

代码示例：

```
//Java code
//set local IP
String localIpAddress = DeviceManager.getLocalIpAddress(false);
TsdkLocalAddress localAddress = new TsdkLocalAddress(localIpAddress);
TsdkManager.getInstance().setConfigParam(localAddress);

TsdkConfAnonymousJoinParam anonymousParam = new TsdkConfAnonymousJoinParam();
anonymousParam.setConfId(joinParam.getAnonymousConfId());
anonymousParam.setConfPassword(joinParam.getConfPassword());
anonymousParam.setDisplayName(joinParam.getDisplayName());
anonymousParam.setServerAddr(joinParam.getServiceAddress());
anonymousParam.setServerPort(Integer.valueOf(joinParam.getServicePort()));
anonymousParam.setUserId(1);
```

eSDK CloudVC 20.1.RC1 开发指南(Android)


```
int result =
TsdkManager.getInstance().getConferenceManager().joinConferenceByAnonymous(anonymousParam);
```

步骤2 SDK通过TsdkNotify对象中的onEvtGetTempUserResult()方法向UI上报临时账号获取结果事件。

说明

若此次事件通知返回失败，应用程序界面应提示用户。

代码示例：

```
//Java code
public void handleGetTempUserResult(int userId, TsdkCommonResult result)
{
    if(result == null){
        return;
    }
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.GET_TEMP_USER_RESULT, result);
}
```

步骤3 SDK通过TsdkNotify对象中的onEvtJoinConfResult()方法向UI上报加入会议结果，后续和正常入会流程一样。

说明

匿名会议过程中，无论用户采用主席密码入会或是普通与会者密码入会，均只有设置自己闭音的会控能力，其他会控能力暂不支持。

代码示例：

```
//Java code
public void handleJoinConfResult(TsdkConference tsdkConference, TsdkCommonResult commonResult,
TsdkJoinConfIndInfo tsdkJoinConfIndInfo
    Log.i(TAG, "handleJoinConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    if (result == 0)
    {
        this.currentConference = tsdkConference;
        this.memberList = null;
        this.self = null;
        TsdkCall tsdkCall = tsdkConference.getCall();
        if (null != tsdkCall) {
            Session newSession =
CallMgr.getInstance().getCallSessionByCallID(tsdkCall.getCallInfo().getCallId());
            if (null == newSession) {
                newSession = new Session(tsdkCall);
                CallMgr.getInstance().putCallSessionToMap(newSession);
            }
            if (tsdkCall.getCallInfo().getIsVideoCall() == 1) {
                VideoMgr.getInstance().initVideoWindow(tsdkCall.getCallInfo().getCallId());
            }
        }
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_CONF_SUCCESS,
tsdkConference.getHandle() + "");
    }
    else
    {
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.JOIN_CONF_FAILED, result);
    }
}
```

----结束

5.5.4 会议控制

5.5.4.1 退出和结束会议

应用场景

普通与会者和主席均可在会议中主动退出会议，主席可以结束会议。

说明

若主席退出会议，则会议中无主席，预约会议时，原主席可以通过主席接入信息重新加入会议获取主席；当会议中无任何与会者时，会议也会自动结束。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

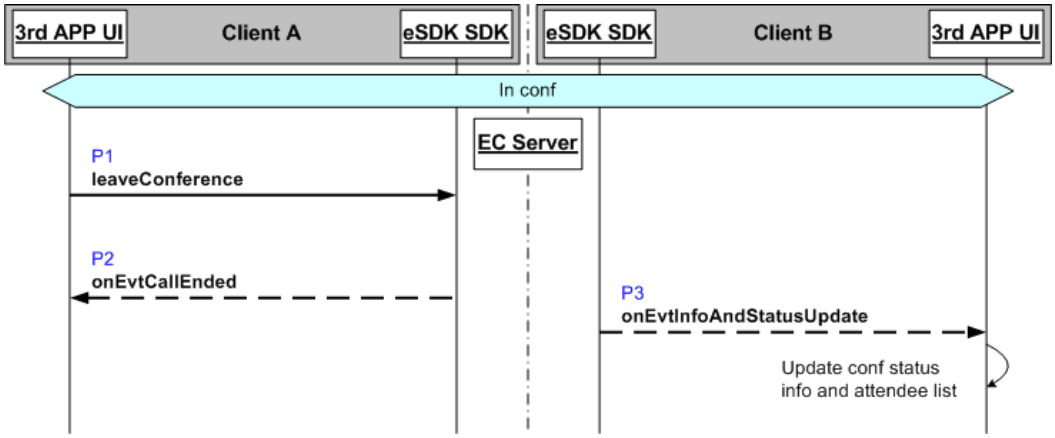
前提条件

与会者已在会议中。

流程说明

退出会议

图 5-25 退出会议流程



步骤1 普通与会者或主席侧UI调用TsdkConference对象中的leaveConference()方法退出会议。

说明

在主动退出会议过程中，sdk层会主动挂断通话，然后才离开会议。

代码示例：

```
//Java code
public int leaveConf()
{
    if (null == currentConference)
    {
        Log.i(TAG,  "leave conf, currentConference is null ");
        return 0;
    }

    int result = currentConference.leaveConference();
    if (result == 0) {
        currentConference = null;
    }

    return result;
}
```

步骤2 普通与会者或主席侧SDK在收到“退出会议”请求的响应后，通过TsdkNotify对象中的OnEvtCallEnded()方法向UI上报呼叫ID销毁事件。

代码示例：

```
//Java code
public void OnEvtCallEnded(TsdkCall call){
    Log.i(TAG, "onCallDestroy");
    if (null == call)
    {
        Log.e(TAG, "call obj is null");
        return;
    }
    Session callSession = getCallSessionByCallID(call.getCallInfo().getCallId());
    if (callSession == null)
    {
        Log.e(TAG, "call session obj is null");
        return;
    }
    removeCallSessionFromMap(callSession);
}
```

步骤3 其他与会者侧SDK收到会议成员列表刷新通知，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及状态更新事件，UI刷新会议状态和成员列表。

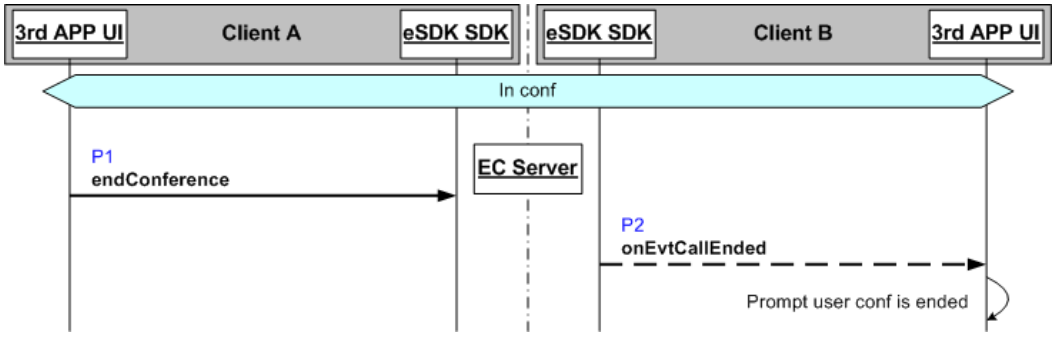
说明

详细流程参见“**会议信息及会议状态更新**”描述。

----结束

结束会议

图 5-26 结束会议流程



步骤1 主席侧UI调用TsdkConference对象中的endConference()方法结束会议。

说明

应用程序界面在关闭会议时应为主席提供“退出会议”和“结束会议”的选择入口。

代码示例：

```
//Java code
public int endConf()
{
    if (null == currentConference)
    {
        Log.i(TAG,  "end conf, currentConference is null ");
        return 0;
    }

    int result =  currentConference.endConference();
    if (result == 0) {
        currentConference = null;
    }

    return result;
}
```

步骤2 其他与会者侧SDK收到会议结束通知，通过TsdkNotify对象中的OnEvtCallEnded()方法向UI上报会议结束通知，UI提示用户会议结束。

代码示例：

```
//Java code
public void handleCallEndedle(TsdkConference conference)
{
    Log.i(TAG, "handleCallEndedle" + conference.getHandle());
    currentConference = null;
}
```

----结束

注意事项

无。

5.5.4.2 基础会控操作

应用场景

在会议中进行基础的会议控制操作。

会控类型	融合会议 Hosted	融合会议 On-premise	备注
闭音会议	Y	Y	
锁定会议	Y	N	
添加与会者	Y	Y	
重拨与会者	N	N	
挂断与会者	Y	Y	
删除与会者	Y	Y	
闭音与会者	Y	Y	
举手	Y(*)	N	Hosted 使用IDo会控协议时暂不支持举手
设置会议视频模式	Y	Y	
选看与会者	Y	Y	
广播与会者	Y	Y	
申请主席	Y	Y	
释放主席	Y	Y	
延长会议	Y	Y	
设置主讲人	Y	Y	
申请主讲人	N	N	
设置会议录播	Y	Y	
设置会议直播	Y	Y	
主席点名与取消	N	Y	
非主席申请发言	N	Y	

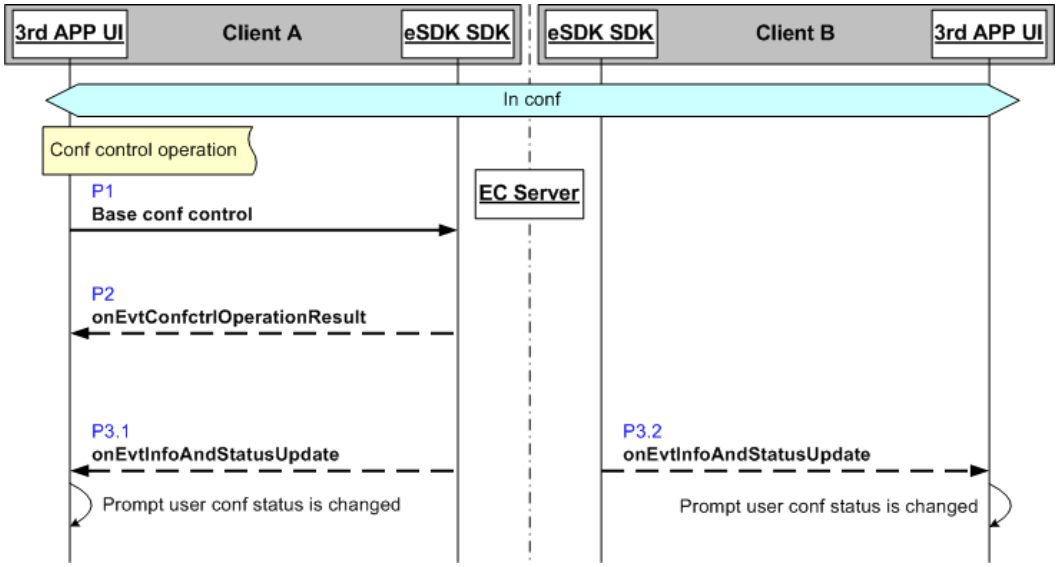
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	eSDK VC 19.1.RC1	CloudVC V600R19C10	新增设置会议录播和直播功能

前提条件

无。

流程说明

图 5-27 基础会控操作流程



步骤1 UI调用TsdkConference对象中的会议基础控制方法（如下表），实现会议控制相关操作。

说明

闭音会场：在Hosted组网下闭音会场，只闭音与会者。在On-premise组网下闭音会场，主席，与会者都闭音。

会控类型	接口	权限	说明
闭音会场	muteConference	主席	设置会议闭音后，除主席外，其他所有与会者均不可说（只可听）
锁定会议	lockConference	主席	会议锁定后，除主席邀请外，其他人不能通过任何途径加入会议
添加与会者	addAttendee	主席	支持邀请一个或多个与会者
重播与会者	redialAttendee	主席	重新呼叫之前挂断的与会者
挂断与会者	handupAttendee	主席	挂断在会议中的与会者
删除与会者	removeAttendee	主席	踢出与会者（正在会议中的）、移除已离会的与会者和取消正在邀请的与会者

会控类型	接口	权限	说明
闭音与会者	muteAttendee	主席 普通与会者	设置闭音后，该与会者不可说(只听)。 会议主席在会议中设置或取消其他与会者闭音，普通与会者设置或取消自己闭音，
举手	setHandup	主席 普通与会者	会议主席在会议中取消其他与会者举手，所有与会者设置或取消自己举手
设置会议视频模式	setVideoMode	主席	在CloudEC解决方案下，支持的视频会议模式有“广播与会者模式”、“声控模式”和“自由讨论模式”
观看与会者	watchAttendee	主席 普通与会者	-
广播与会者	broadcastAttendee	主席	会议视频模式为“广播与会者模式”时主席可以指定广播与会者
申请主席	requestChairman	普通与会者	通过主席密码申请主席权限成为主席
释放主席	releaseChairman	主席	释放主席权限后成为普通与会者
设置会议录播	setRecordBroadcast	主席	设置开始和结束录播
设置会议直播	setLiveBroadcast	主席	设置开始和结束直播
延长会议	postpone	主席	设置延长会议时间
主席点名与取消	rollCallAttendee	主席	设置主席点名普通与会者发言与取消点名
非主席申请发言	requestSpeaking	普通与会者	

代码示例：

```
//Java code
public int muteConf(boolean isMute)
{
    if (null == currentConference)
    {
        Log.e(TAG,  "mute conf failed, currentConference is null ");
        return -1;
    }

    int result =  currentConference.muteConference(isMute);

    return result;
}
```


步骤2 SDK在收到会控请求的响应后，通过TsdkNotify对象中的onEvtConfctrlOperationResult()方法向UI上报会控操作结果事件。

代码示例：

```
//Java code
public void handleConfctrlOperationResult(TsdkConference conference, TsdkConfOperationResult result)
{
    Log.i(TAG, "handleConfctrlOperationResult");
    int ret = result.getReasonCode();
    if (null == conference || null == result)
    {
        return;
    }
    if (ret != 0)
    {
        Log.e(TAG, "conf ctrl operation failed: " + result.getDescription());
        return;
    }
    int confOperationType = result.getOperationType();
    switch (confOperationType)
    {
        case 2:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.MUTE_CONF_RESULT, ret);
            break;
        case 3:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.UN_MUTE_CONF_RESULT, ret);
            break;
        case 4:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.LOCK_CONF_RESULT, ret);
            break;
        case 5:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.UN_LOCK_CONF_RESULT, ret);
            break;
        case 6:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.ADD_ATTENDEE_RESULT, ret);
            break;
        case 7:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.DEL_ATTENDEE_RESULT, ret);
            break;
        case 10:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.MUTE_ATTENDEE_RESULT, ret);
            break;
        case 11:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.UN_MUTE_ATTENDEE_RESULT,
ret);
            break;
        case 12:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.HAND_UP_RESULT, ret);
            break;
        case 13:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.CANCEL_HAND_UP_RESULT, ret);
            break
        case 18:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.REQUEST_CHAIRMAN_RESULT,
ret);
            break;
        case 19:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.RELEASE_CHAIRMAN_RESULT,
ret);
            break;
        default:
            break;
    }
}
```

步骤3 所有会议成员侧SDK收到会议成员列表刷新通知，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及状态更新事件，UI刷新会议状态和成员列表。

说明

详细流程参见“**会议信息及会议状态更新**”描述。

----结束

注意事项

无。

5.5.4.3 会议信息及会议状态更新

应用场景

会议过程中，会议状态或与会者成员状态发生变化时，服务器会推送变更通知，应用程序界面应刷新相应的状态以提示用户。

说明

1、会议状态当前包括锁定状态、闭音状态和是否录播状态；
2、会议成员状态当前包括与会者加入、退出、挂断、闭音、举手和角色变更。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

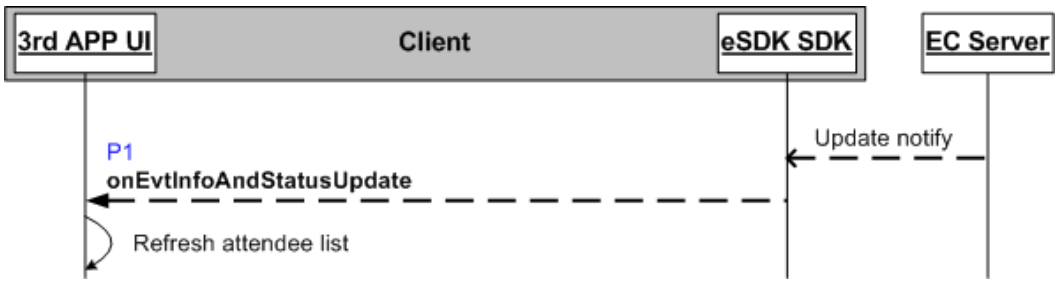
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	eSDK VC 19.1.RC1	CloudVC V600R19C10	会议信息新增录播参数

前提条件

主席和与会者均已在会议中。

流程说明

图 5-28 更新会议状态信息和与会者列表流程



eSDK CloudVC 20.1.RC1 开发指南(Android)

步骤1 SDK在收到服务器返回的会议状态信息和与会者列表更新通知后，通过TsdkNotify对象中onEvtInfoAndStatusUpdate()方法向UI上报会议信息及会议状态更新事件。

说明

事件返回的消息内容为TsdkConference对象，其中：

- 会场录音状态(record)，为true时，应用程序界面应提示与会者当前处于录音状态；
- 会场锁定状态(lock)，为true时，应用程序界面应提示与会者当前处于锁定状态；
- 会场静音状态(allMute)，为true时，应用程序界面应提示与会者当前处于会场闭音状态，除主席外，均可听不可说；
- 会议主题(subject)，不为空时，应用程序界面应显示此主题，在会议创建时已确定，首次获取后，不会再有变更；
- 成员更新方式(confAttendeeUpdateType)参考TsdkConfAttendeeUpdateType，支持无更新、全量更新、增量增加、增量修改和增量删除，目前以全量同步的方式进行更新；
- 是否高清视频会议(hdConf)，为true时，会议视频为高清视频；
- 会议媒体类型(confMediaType)，参考TsdkConfMediaType枚举类；
- 会议状态(confState)，参考TsdkConfState枚举类；
- 与会者个数，当前会议中的与会者个数可通过getAttendeeList().size()获取；
- 与会者的详细信息(attendeeList)，包括用户标识、名称、号码、闭音状态、静音状态、举手状态、用户状态、角色和支持的媒体类型，应用程序界面需要根据相应字段显示与会者的信息和状态；
- 是否支持直播（supportLiveBroadcast），为true时，支持直播；
- 是否直播状态（liveBroadcast），为true时，直播状态；
- 是否支持录播（supportRecordBroadcast），为true时，支持录播。

备注：离会，删除与会者，退出会议等情景中 如遇不知如何判断是否还在会议中时 Android平台下都可以通过刷新事件判断自己是否在列表中来判断自己的状态。

代码示例：

```
//Java code
public void handleInfoAndStatusUpdate(TsdkConference conference){
    Log.i(TAG, "onConfStatusUpdateInd");
    if ((currentConference == null) || (conference == null))
    {
        return;
    }
    if (currentConference.getHandle() != conference.getHandle())
    {
        return;
    }
    String handle = conference.getHandle()+"";
    this.updateConfInfo(conference);
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.STATE_UPDATE, handle);
}
```

----结束

注意事项

无。

5.5.4.4 显示发言人

应用场景

会议过程中，应用程序显示服务器推送的当前发言人信息。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

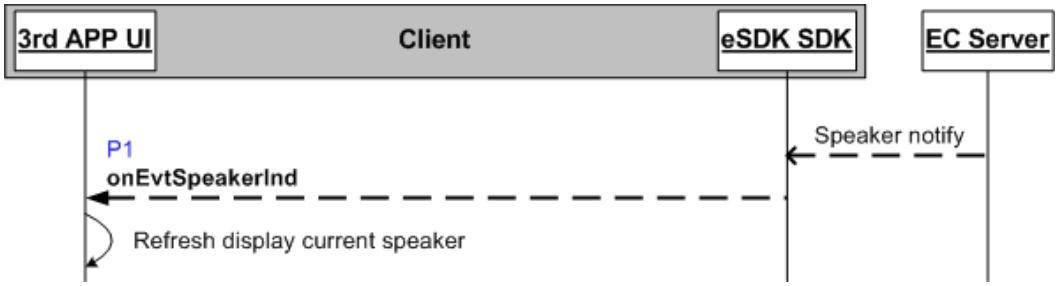
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 6.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

与会者均已在会议中。

流程说明

图 5-29 显示发言人流程



步骤1 SDK在收到服务器返回的发言人通知后，通过TsdkNotify对象中onEvtSpeakerInd()方法向UI上报发言人通知事件，携带当前发言方信息以及会议信息。

说明

当存在多个发言人时，建议应用程序界面按音量大小，显示第一发言人和第二发言人。

----结束

注意事项

无。

5.5.5 渐进式会议

5.5.5.1 两方通话转会议

应用场景

用户在点对点通话过程中发起转会议，或邀请第三方（或更多）加入通话。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

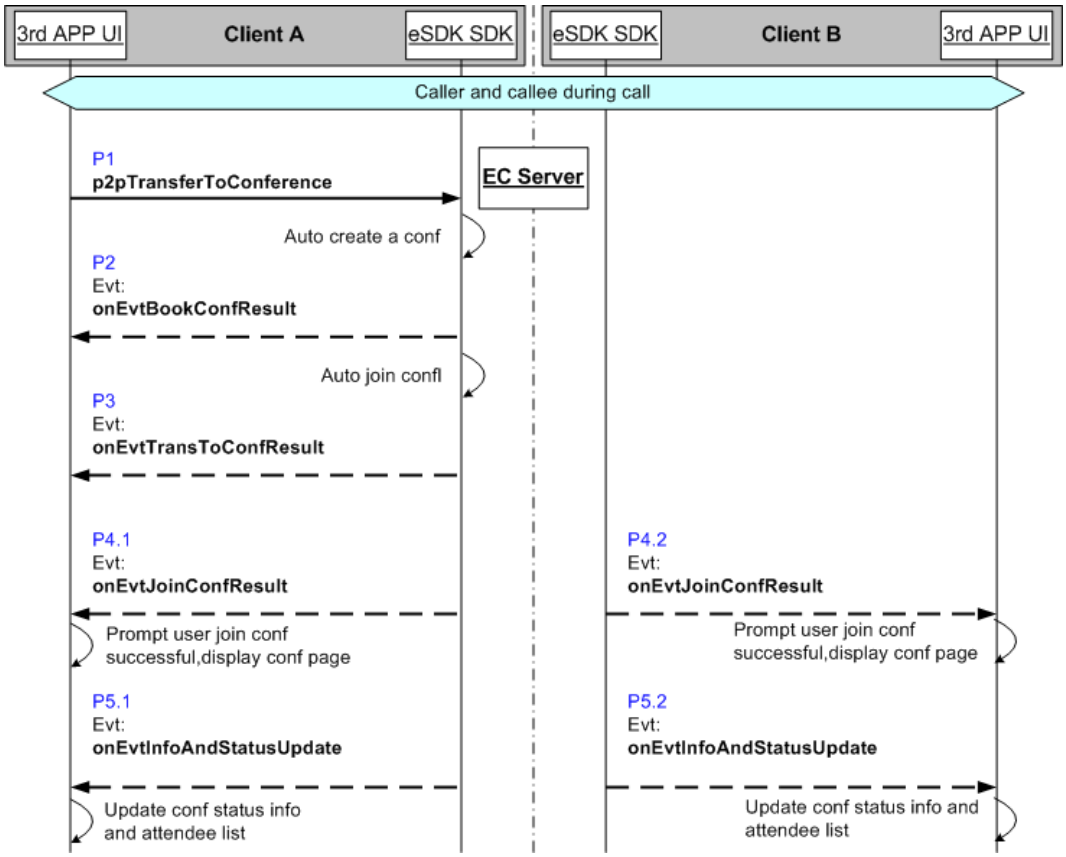
前提条件

1. 鉴权登录成功；
2. SIP号码已成功注册；
3. 会议环境参数已设置。

流程说明

两方通话转会议

图 5-30 两方通话转会议流程



步骤1 会议发起方UI调用TsdkConferenceManager对象中的p2pTransferToConference()方法发起通话转会议，参数为TsdkBookConfInfo类和TsdkCall类。

代码示例：

```
//java code
public int callTransferToConference(int call_id){

    Log.i(TAG, "callTransferToConference.");

    Session callSession = CallMgr.getInstance().getCallSessionByCallID(call_id);
    if (callSession == null)
    {
        Log.e(TAG, "call Session is null.");
        return -1;
    }

    TsdkCall tsdkCall =  callSession.getTsdkCall();
    if (tsdkCall == null)
    {
        Log.e(TAG, "call is invalid.");
        return -1;
    }

    TsdkBookConfInfo bookConfInfo = new TsdkBookConfInfo();

    bookConfInfo.setConfType(TsdkConfType.TSDK_E_CONF_INSTANT);
    bookConfInfo.setIsAutoProlong(1);

    bookConfInfo.setSubject(LoginMgr.getInstance().getAccount() + "'s Meeting");
    if (1 == tsdkCall.getCallInfo().getIsVideoCall()) {
        bookConfInfo.setConfMediaType(TSDK_E_CONF_MEDIA_VIDEO);
    }else {
```

```
        bookConfInfo.setConfMediaType(TSDK_E_CONF_MEDIA_VOICE);
    }

    bookConfInfo.setSize(2);

    List<TsdkAttendeeBaseInfo> attendeeList = new ArrayList<>();
    TsdkAttendeeBaseInfo confctrlAttendee = new TsdkAttendeeBaseInfo();
    //confctrlAttendee.setNumber(callInfo.getPeerNumber());
    confctrlAttendee.setRole(TsdkConfRole.TSDK_E_CONF_ROLE_ATTENDEE);
    attendeeList.add(confctrlAttendee);

    bookConfInfo.setAttendeeList(attendeeList);
    bookConfInfo.setAttendeeNum(attendeeList.size());

    //The other parameters are optional, using the default value
    //其他参数可选，使用默认值即可
    bookConfInfo.setLanguage(TsdkConfLanguage.TSDK_E_CONF_LANGUAGE_EN_US);

    int result = TsdkManager.getInstance().getConferenceManager().p2pTransferToConference(tsdkCall,
bookConfInfo);
    if (result != 0) {
        Log.e(TAG, "call transfer to conference is return failed, result = " + result);
    }
    return result;
}
```

步骤2 发起方SDK在收到服务器返回的立即会议创建响应后，通过TsdkNotify对象中的onEvtBookConfResult()方法向向UI上报会议创建结果。

步骤3 会议发起方SDK完成通话转会议处理，自动挂断原通话，通过TsdkNotify对象中的onEvtTransToConfResult()方法向UI上报通话转会议结果，携带原通话对象。

说明

若转会议失败，原通话仍存在，UI需要使用原通话对象恢复原通话界面；若转会议成功，原通话对象由SDK自动回收，此时返回的原通话对象可能为空。

步骤4 通话双方SDK完成通话转会议处理，通过TsdkNotify对象中的onEvtJoinConfResult()方法加入会议结果，并返回会议对象TsdkConference，后续会控时使用。此时UI可跳转至会议界面。

步骤5 所有与会者侧SDK收到会议状态刷新通知，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及会议状态刷新事件，UI刷新会议状态和成员列表。

----结束

注意事项

无。

5.5.5.2 升级会议

应用场景

- “语音会议”升级至“语音+数据会议”。
- “视频会议”升级至“视频+数据会议”。
- 暂不支持“语音会议”升级至“语音+视频会议”。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	N	

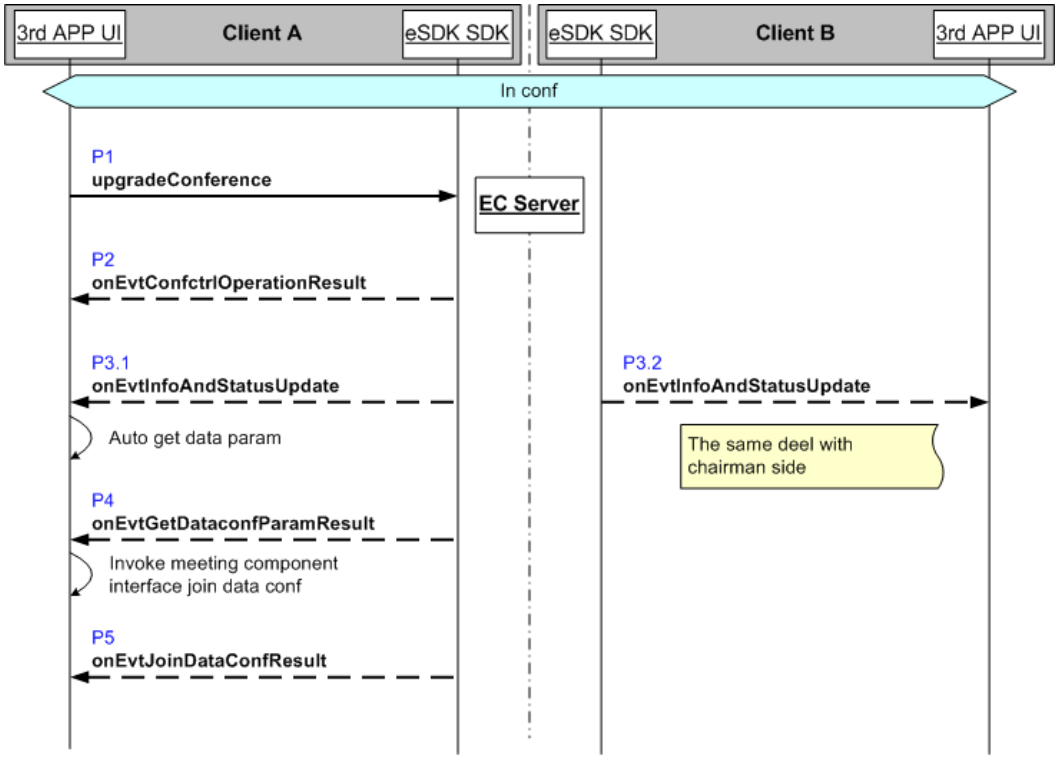
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

与会者通过主席身份加入会议，或已申请获取为主席。

流程说明

图 5-31 升级会议流程



步骤1 会议发起方UI调用TsdkConference对象中的upgradeConference()方法升级为数据会议，参数为群组ID，SDK发送升级会议请求至服务器。

代码示例：


```
//Java code
public int upgradeConf()
{
    if (null == currentConference)
    {
        Log.e(TAG,  "upgrade conf failed, currentConference is null ");
        return -1;
    }

    int result =  currentConference.upgradeConference("");
    return result;
}
```

步骤2 发起方SDK在收到“升级为数据会议”请求的响应后， 通过TsdkNotify对象中的onEvtConfctrlOperationResult()方法向UI上报升级会议的会控操作结果。

代码示例：

```
//Java code
public void handleConfctrlOperationResult(TsdkConference conference, TsdkConfOperationResult result)
{
    Log.i(TAG, "handleConfctrlOperationResult");
    int ret = result.getReasonCode();
    if (null == conference || null == result)
    {
        return;
    }
    if (ret != 0)
    {
        Log.e(TAG, "conf ctrl operation failed: " + result.getDescription());
        return;
    }
    int confOperationType = result.getOperationType();
}
```

步骤3 会议发起方、接收方SDK收到会议成员列表刷新通知，通过TsdkNotify对象中的onEvtInfoAndStatusUpdate()方法向UI上报会议信息及状态刷新事件，UI刷新会议状态和成员列表。

说明

详细流程参见“**会议信息及会议状态更新**”描述。

步骤4 SDK自动获取数据会议入会参数，通过TsdkNotify对象中的onEvtGetDataconfParamResult()方法向UI上报数据会议入会参数获取结果，UI应调用TsdkConference对象中的joinDataConference()方法加入数据会议。

代码示例：

```
//Java code
public void  handleGetDataConfParamsResult(TsdkConference tsdkConference, TsdkCommonResult
commonResult){
    Log.i(TAG, "handleJoinConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.GET_DATA_CONF_PARAM_RESULT,
result);
}
```

步骤5 SDK自动获取数据会议入会参数，通过TsdkNotify对象中的onEvtJoinDataConfResult()方法向UI上报加入数据会议结果，UI根据结果显示相应标识按钮。

代码示例：

```
//Java code
public void  handleGetDataConfParamsResult(TsdkConference tsdkConference, TsdkCommonResult
```

```
commonResult){
    Log.i(TAG, "handleJoinConfResult");
    if ((tsdkConference == null) || (commonResult == null)) {
        return;
    }
    int result = commonResult.getResult();
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.GET_DATA_CONF_PARAM_RESULT,
result);
}
```

----结束

注意事项

无。

5.5.6 桌面协同与共享

5.5.6.1 屏幕和程序共享

应用场景

会议中，移动与会者观看屏幕或程序共享。

说明

移动应用程序暂不具备共享屏幕和程序的能力，屏幕或程序的共享者为PC应用程序。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

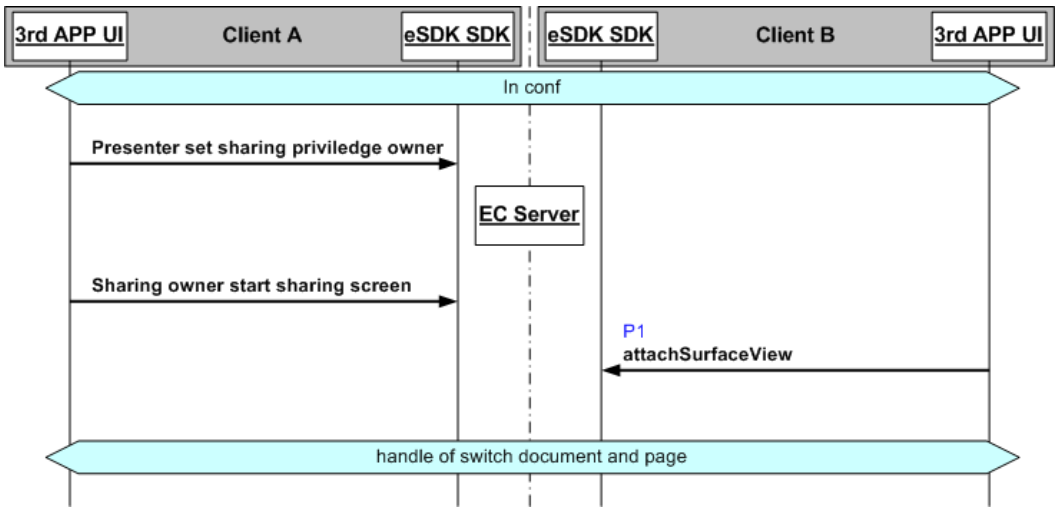
1、加入数据会议成功；

2、加载屏幕/程序共享模块成功。

流程说明

开始观看屏幕或程序共享

图 5-32 开始观看屏幕或程序共享流程



说明

1. 会议中，主讲人设置共享权限拥有者，邀请其他与会者进行屏幕或程序共享。
2. 共享者开始共享屏幕或程序。

步骤1 UI调用TsdkConference类的attachSurfaceView()方法绑定视图，刷新共享显示区域。

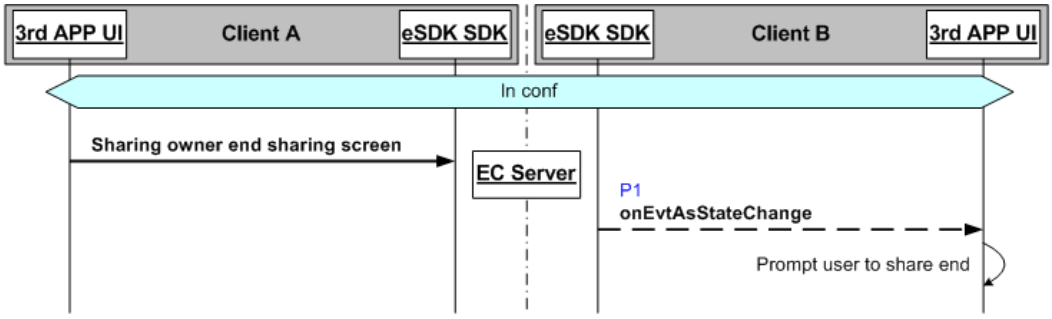
代码示例：

```
//Java code
public void attachSurfaceView(ViewGroup container, Context context)
{
    if (null == currentConference)
    {
        Log.e(TAG,  "attach surface view failed, currentConference is null ");
        return;
    }
    currentConference.attachSurfaceView(container, context);
}
```

----结束

结束观看屏幕或程序共享

图 5-33 结束观看屏幕或程序共享流程



说明

1. PC端用户在进行屏幕或者程序共享中结束共享屏幕或程序。

步骤1 SDK在收到服务器返回的结束共享状态通知后，通过TsdkNotify对象中onEvtAsStateChange()方法向UI上报应用共享状态通知事件。

说明

共享状态可参考枚举类TsdkConfShareState，其中TsdkConfShareState.TSDK_E_CONF_AS_STATE_NULL为结束程序或者屏幕共享状态。

代码示例：

```
//Java code
public void handleAsStateChange(TsdkConfAsStateInfo asStateInfo)
{
    switch (asStateInfo.getState())
    {
        case 0:
            mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.END_AS_SHARE, asStateInfo);
            break;
        default:
            break;
    }
}
```

----结束

注意事项

无。

5.5.6.2 文档和白板共享

应用场景

会议中，移动与会者观看文档或白板共享。

说明

移动应用程序暂不具备共享文档和白板的能力，文档或白板的共享者为PC应用程序。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

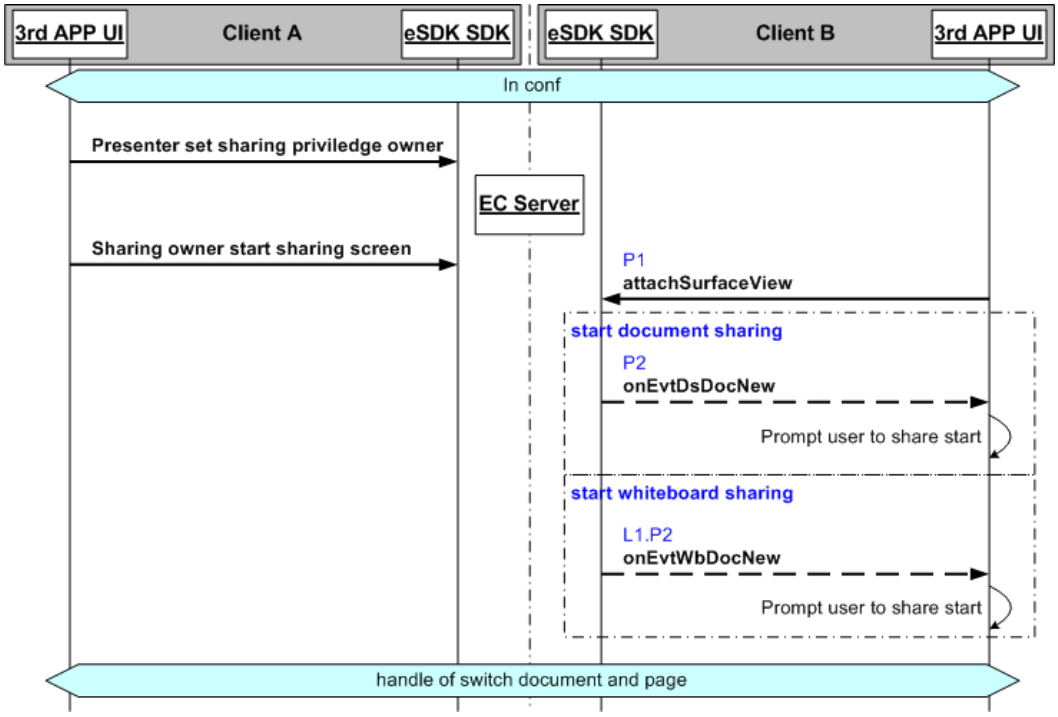
1、加入数据会议成功；

2、加载文档/白板共享模块成功。

流程说明

开始观看文档或白板共享

图 5-34 开始观看文档或白板共享流程



说明

1. 会议中，主讲人设置共享权限拥有者，邀请其他与会者进行文档或白板共享。
2. 共享者开始共享文档或白板。

步骤1 UI调用TsdkConference类的attachSurfaceView()方法绑定视图，刷新共享显示区域。

代码示例：

```
//Java code
public void attachSurfaceView(ViewGroup container, Context context)
{
    if (null == currentConference)
    {
        Log.e(TAG,  "attach surface view failed, currentConference is null ");
        return;
    }
    currentConference.attachSurfaceView(container, context);
}
```

步骤2 如果文档共享者打开文档，移动与会者侧SDK下载到一个文档头后，通过TsdkNotify对象中onEvtDsDocNew()方法向UI上报新建一个共享文档事件；如果共享侧新建一个空的白板文档，移动侧SDK下载到一个白板文档头后，通过TsdkNotify对象中onEvtWbDocNew()方法向UI上报新建一个白板文档事件。

代码示例：

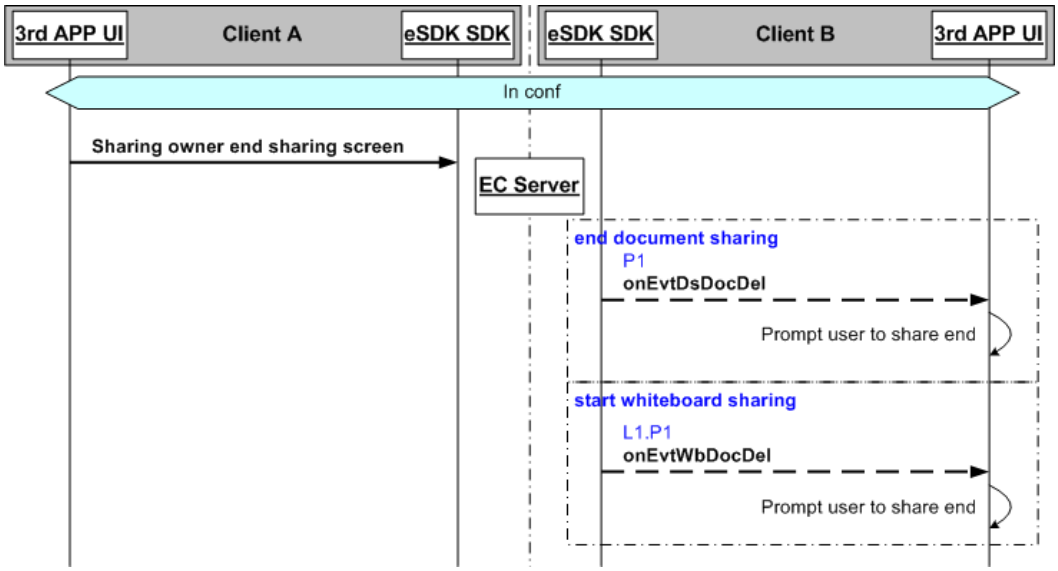
```
Java code
public void handleWbDocNew(TsdkDocBaseInfo docBaseInfo)
{
    if (null == docBaseInfo)
    {
        return;
    }

    documentId.add(docBaseInfo.getDocumentId());
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.START_DATA_CONF_SHARE,
docBaseInfo);
}
```

```
Java code
public void handleDsDocNew(TsdkDocBaseInfo docBaseInfo)
{
    if (null == docBaseInfo)
    {
        return;
    }

    documentId.add(docBaseInfo.getDocumentId());
    mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.START_DATA_CONF_SHARE,
docBaseInfo);
}
```

----结束

结束观看文档或白板共享

图 5-35 结束观看文档或白板共享流程



说明

1. PC端用户在进行文档或者白板共享中结束共享文档或白板。

步骤1 如果移动侧SDK通过TsdkNotify对象中的onEvtDsDocDel()方法向UI上报文档被删除通知消息，UI关闭文档共享显示窗口；如果移动侧SDK通过TsdkNotify对象中的onEvtWbDocDel()方法向UI上报白板被删除通知消息，UI关闭白板共享显示窗口。

代码示例：

```
//Java code
public void handleDsDocDel(TsdkDocShareDelDocInfo docShareDelDocInfo)
{
    if (null == docShareDelDocInfo)
    {
        return;
    }

    Iterator<Integer> iterator = documentId.iterator();
    while (iterator.hasNext())
    {
        if (iterator.next() == docShareDelDocInfo.getDocBaseInfo().getDocumentId())
        {
            iterator.remove();
        }
    }

    if (0 == documentId.size() && !isShareAs)
    {
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.END_DATA_CONF_SHARE,
docShareDelDocInfo);
    }
}
```

```
//Java code
public void handleWbDocDel(TsdkWbDelDocInfo wbDelDocInfo)
{
    if (null == wbDelDocInfo)
    {
        return;
    }

    Iterator<Integer> iterator = documentId.iterator();
    while (iterator.hasNext())
    {
        if (iterator.next() == wbDelDocInfo.getWbBaseInfo().getDocumentId())
        {
            iterator.remove();
        }
    }

    if (0 == documentId.size() && !isShareAs)
    {
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.END_DATA_CONF_SHARE,
wbDelDocInfo);
    }
}
```

----结束

注意事项

无。

5.5.6.3 聊天

应用场景

会议中，所有人可以收到其他与会者发送的聊天消息内容。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 On-premise	Y	

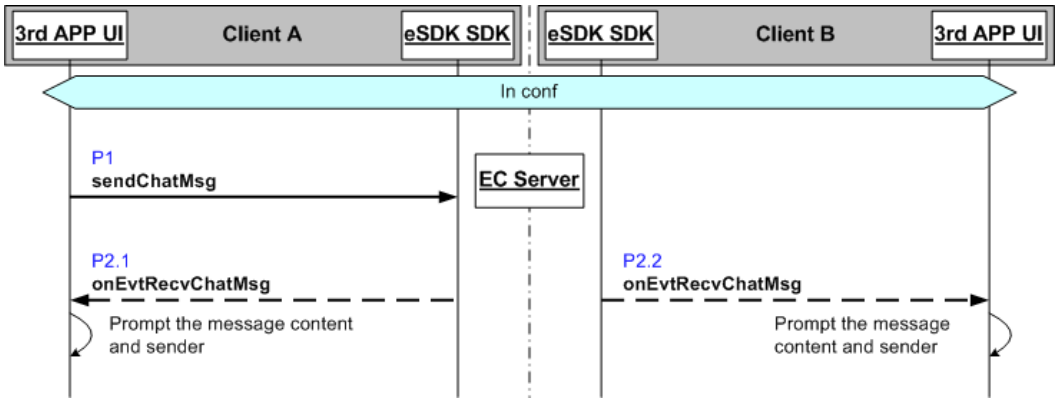
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

1、加入数据会议成功。

流程说明

图 5-36 发送聊天消息



步骤1 UI调用TsdkConference类的sendChatMsg()方法发送聊天消息。

代码示例：

```
//Java code
public void sendConfMessage(String message)
   {
      if (null == currentConference)
      {
         LogUtil.e(TAG,  "send chat failed, currentConference is null ");
         return;
      }
      TsdkConfChatMsgInfo chatMsgInfo = new TsdkConfChatMsgInfo();
      chatMsgInfo.setChatType(TsdkConfChatType.TSDK_E_CONF_CHAT_PUBLIC);
      chatMsgInfo.setChatMsg(message);
      if (null == self)
      {
         chatMsgInfo.setSenderDisplayName(LoginMgr.getInstance().getAccount());
      }
      else
      {
```

```
            chatMsgInfo.setSenderDisplayName(self.getDisplayName());
        }
        currentConference.sendChatMsg(chatMsgInfo);
    }
```

步骤2 发送消息侧、其他与会者侧SDK收到服务器返回的发送消息响应，向UI上报会议中的聊天消息通知事件onEvtRecvChatMsg，UI刷新界面显示消息内容。

代码示例：

```
//Java code
public void handleRecvChatMsg(TsdkConfChatMsgInfo confChatMsgInfo)
    {
        mConfNotification.onConfEventNotify(ConfConstant.CONF_EVENT.CONF_CHAT_MSG,
confChatMsgInfo);
    }
```

----结束

注意事项

无。

5.5.6.4 通用消息通道

应用场景

会议过程中，与会者可以使用通用消息通道发送用户自定义数据消息。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

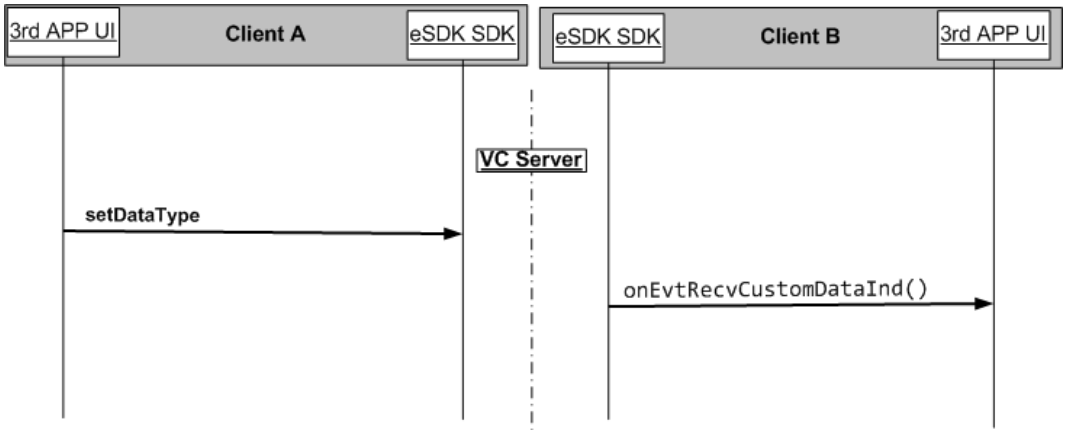
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

1. 数据会议且与会者均已在会议中。

流程说明

图 5-37 发送和接收公共消息流程



步骤1 UI调用接口setDataType()接口在会议中发送用户自定义数据。

代码示例：

```
public void sendConfData(int dataType, String message) {
        if (null == currentConference) {
            LogUtil.e(TAG, "send chat failed, currentConference is null ");
            return;
        }
        LogUtil.i(TAG,"MeetingMgr sendConfData:"+message.toString());

        TsdkConfChatMsgInfo chatMsgInfo = new TsdkConfChatMsgInfo();
        chatMsgInfo.setDataType(dataType);
        chatMsgInfo.setChatMsg(message);
        if (null == self)
        {
            chatMsgInfo.setSenderDisplayName(LoginMgr.getInstance().getAccount());
        }
        else
        {
            chatMsgInfo.setSenderDisplayName(self.getDisplayName());
        }
        LogUtil.i(TAG,"chatMsgInfo.getDataType():"+chatMsgInfo.getDataType()+"
chatMsg:"+chatMsgInfo.getChatMsg().toString());
        currentConference.sendConfData(chatMsgInfo);
    }
```

步骤2 SDK向UI上报onEvtRecvCustomDataInd事件，UI解析数据并显示，发送公共数据时发送方收不到，其他与会者都能收到，发送个人数据时，只有接收方能收到数据。

代码示例：

```
case
public void onEvtRecvCustomDataInd(TsdkOnEvtRecvCustomDataInd notify) {}
```

----结束

注意事项

无。

5.5.7 观看辅流共享

应用场景

会议中，移动与会者观看辅流共享。

说明

移动应用程序暂不具备共享屏幕和程序的能力，屏幕或程序的共享者为PC应用程序。

组网	是否支持（Y:支持 N:不支持）	备注
融合会议 Hosted	Y	
融合会议 On-premise	Y	

变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 20.1.0	CloudVC V600R020C10	
变更	NA	NA	NA

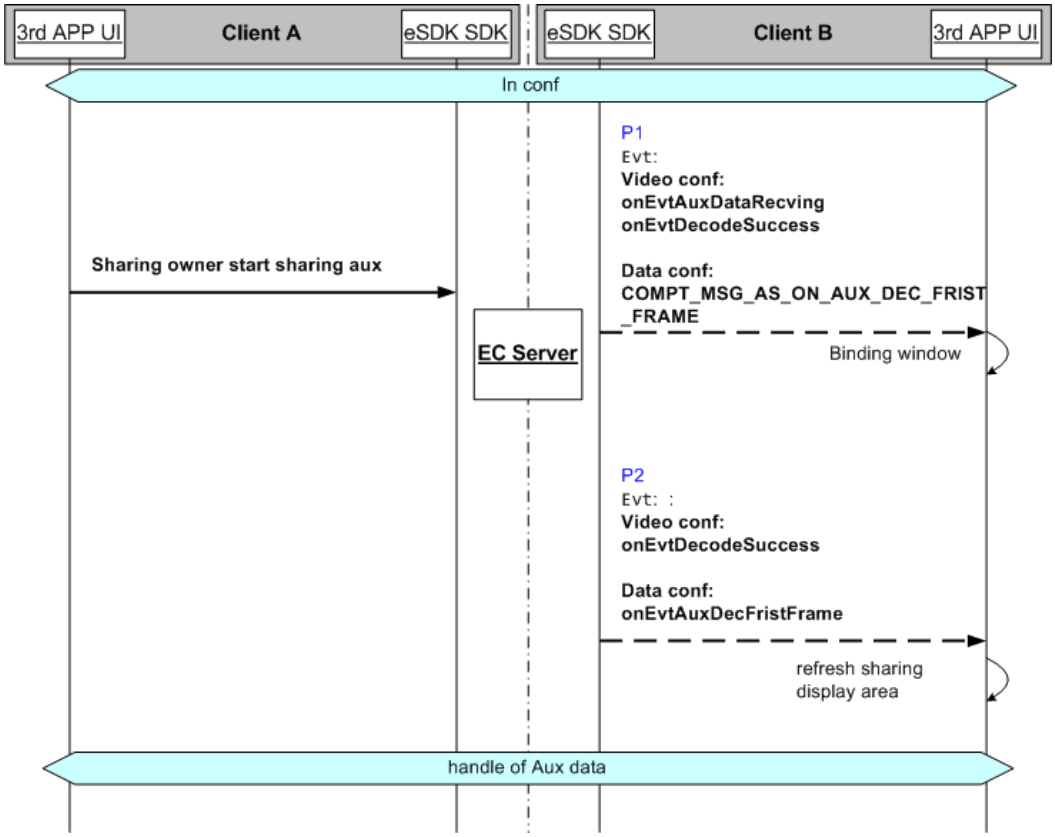
前提条件

1. 加入视频会议或数据会议成功；
2. 加载辅流共享模块成功。

流程说明

开始观看辅流共享

图 5-38 开始观看辅流共享流程



说明

视频会议需要调用接口setVideoWindow来设置绑定辅流窗口，数据会议不需要。

步骤1 要前期将辅流窗口创建，保存辅流窗口的下标，在调用接口setVideoWindow绑定本地和远端窗口的同时绑定设置辅流窗口。

代码示例：

```
//Java code
/**
 * 辅流窗口（只能创建一个，创建方法和远端窗口一致）
 */
 private SurfaceView auxDataView;
// 创建辅流窗口
auxDataView = ViERenderer.createRenderer(context, true);
auxDataView.setZOrderMediaOverlay(true);
//保存辅流窗口的Index
int bfcpCallIndex = ViERenderer.getIndexOfSurface(auxDataView);
TsdkManager.getInstance().getCallManager().setBfcpCallIndex(bfcpCallIndex);
LogUtil.i(TAG, "auxDataView bfcpCallIndex： " + bfcpCallIndex);

//设置辅流窗口
TsdkVideoWndInfo auxDataWndInfo = new TsdkVideoWndInfo();
auxDataWndInfo.setVideoWndType(TsdkVideoWndType.TSDK_E_VIDEO_WND_AUX_DATA);
auxDataWndInfo.setRender(ViERenderer.getIndexOfSurface(auxDataView));
auxDataWndInfo.setDisplayMode(TsdkVideoWndDisplayMode.TSDK_E_VIDEO_WND_DISPLAY_CUT);

List<TsdkVideoWndInfo> list = new ArrayList<>();
list.add(localWndInfo);
list.add(remoteWndInfo);
list.add(auxDataWndInfo);
```

```
TsdkCall tsdkCall = callManager.getCallByCallId(callId);
if (tsdkCall != null) {
    tsdkCall.setVideoWindow(list);
}
```

步骤2 共享者开始共享辅流，移动与会者侧SDK上报共享辅流通知消息，对应的消息ID为：

视频会议：onEvtAuxDataRecving（视频辅流接收通知），onEvtDecodeSuccess（视频辅流解码成功的通知）。

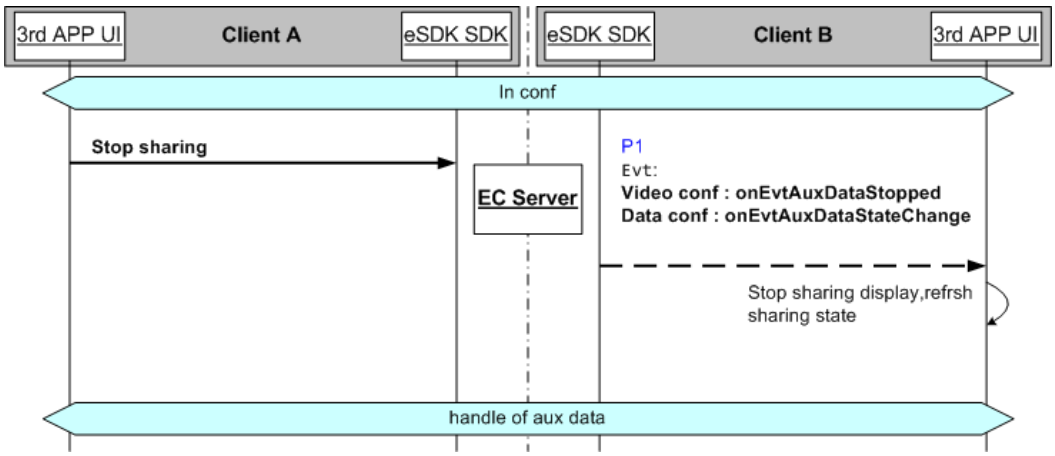
数据会议：onEvtAuxDecFristFrame（辅流第一帧通知）。

步骤3 收到onEvtDecodeSuccess和onEvtAuxDecFristFrame两个通知后展示绑定的辅流界面，刷新UI。

----结束

辅流共享结束处理

图 5-39 辅流共享结束处理流程



步骤1 共享者停止共享，移动与会者侧SDK通过上报辅流共享结束通知消息，对应的消息ID为：

视频会议：onEvtAuxDataStopped（视频辅流停止通知）

数据会议：onEvtAuxDataStateChange（辅流绑定和解绑通知），目前只有解绑的状态上报，即onEvtAuxDataStateChange的参数auxDataStateChange为false，UI刷新界面结束共享。

----结束

注意事项

无

5.6 企业通讯录

应用场景

用户查询企业通讯录。

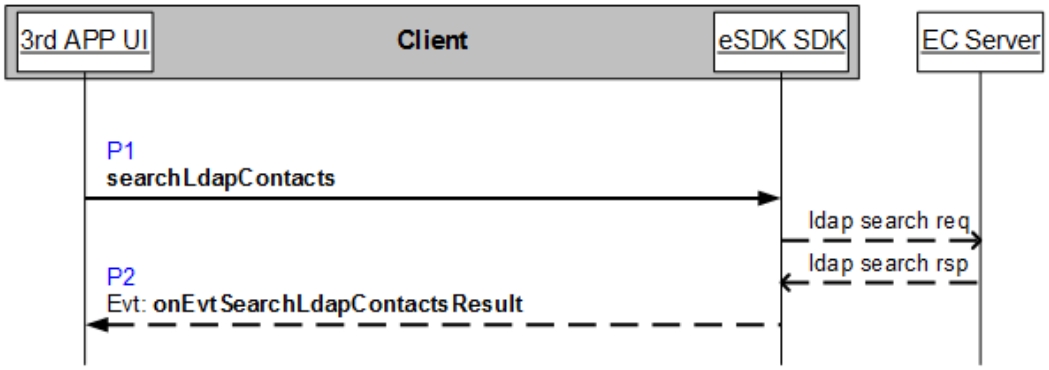
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 20.1.0	CloudVC V600R020C10	
变更	NA	NA	NA

前提条件

1. 鉴权登录成功；
2. SMC上配置企业通讯录服务器对接正常。

流程说明

图 5-40 查询企业通讯录流程



步骤1 UI调用TsdkLdapFrontstageManager对象中的searchLdapContacts()发起查询，参数结构为TsdkSearchLdapContactsParam，SDK发送查询请求至EUA服务器。

说明

1. 在查询企业通讯录时，其中参数TsdkSearchLdapContactsParam结构中各个字段取值如下：
   1. keywords为空字符串时代表查询特性组织结构下所有子级组织结构和所有联系人，为*时代表查询所有联系人；
   2. pageSize为0时代表不分页，为大于0的值时，该值代表每页返回的联系人数，最大为300；
   3. searchSingleLevel为查询分组的级别，

      是否搜索特定分组下的地址本。取值如下：

      0：根目录全量搜索

      1：特定目录全量搜索 返回这个节点下面的所有内容(子节点的子节点等)

      2：特定目录单级搜索 只返回这个节点的子节点内容
   4. currentBaseDN代表在指定的组织结构下查询联系人，为空字符串时，代表在所有组织结构下查询；组织结构的格式如：ou=VC, cd=Huawei, cd=com代表在VC这个组织结构下查询；ou=test, ou=VC, cd=Huawei, cd=com代表在VC部门下的test部门下查询。可从查询返回的联系人结构TsdkLdapContactsInfo中的corpName_字段获取。
   5. cookieLength和pageCookie参数在分页查询时，第一次查询时cookieLength为0、pageCookie为空字符串，在后续分页查询时为上次返回的查询结果TsdkOnEvtSearchLdapContactsResult.Param结构中的cookieLen和pageCookie。
2. 用户登录成功后，会收到服务单推送过来的EUA服务器信息，eSDK会启动查询服务，只有在查询服务开启后，才能进行查询。

代码示例：

```
None
```

步骤2 SDK在收到服务器返回的查询结果响应后，向UI上报企业通讯录查询结果通知onEvtSearchLdapContactsResult，对应的结果数据结构为TsdkOnEvtSearchLdapContactsResult.Param。

说明

1. 如果企业通讯录查询成功，UI层获取到联系人信息，使用联系人的账号进行呼叫、预约会议等操作。
2. UI层收到企业通讯录查询结果通知后，需要缓存TsdkOnEvtSearchLdapContactsResult.Param结构中的内容，eSDK在执行完界面层的回调后，会释放通知中的data指针的内存。
3. 如果查询TsdkSearchLdapContactsParam结构体中的keywords为空字符串，返回的TsdkOnEvtSearchLdapContactsResult.Param结构中，可能会包含currentBaseDN组织结构下的子级组织结构。组织结构和联系人的数据都使用TsdkLdapContactsInfo结构体，当TsdkLdapContactsInfo结构体表示组织结构是，只有corpName字段、deptName字段和webSite字段有值，其他字段都为空字符串。
4. UI层可根据返回的组织结构，构建界面上的组织结构树。TsdkLdapContactsInfo结构体corpName字段格式如："ou=test, ou=VC, cd=Huawei, cd=com"，其中包含多个ou，第一个ou值代表最小组织结构，后面的ou值代表前面ou值的父级组织结构，以此类推。

代码示例：

```
None
```

----结束

注意事项

无。

eSDK CloudVC 20.1.RC1 开发指南(Android)

6 问题定位

6.1 错误码分析

获取错误码

- **接口调用返回**

  每个接口调用后，无论调用成功还是失败，都会有一个返回值。

  返回值类型为INT32/UINT32/LONG的接口，如果返回值为0，表示成功；否则表示失败，该返回值即为错误码。

  返回值类型为BOOL/bool的接口，如果返回值为TRUE/true/YES，表示成功，否则表示失败。

  返回值类型为对象或指针的，如果返回值非空，表示成功，否则表示失败。

  说明

  在SDK中，大量接口实际上是一个异步调用接口，接口返回成功，只是表明接口的调用成功，并不表示业务执行成功。

- **回调通知返回**

  SDK的业务执行结果类回调通知，返回实际的业务执行结果，如果返回值为0，表示成功；否则表示失败，该返回值即为错误码。

- **日志获取**

  日志文件里记录了业务接口调用的时间、接口入参(非用户敏感信息关键参数)以及调用结果。

错误码分析

在接口参考文档里，列出了所有错误码信息，开发者可根据接口返回的错误码，查询相对应的错误描述。

表 6-1 模块错误码定义

模块	相关错误码或原因值定义	说明
初始化	TsdkManagerErrId	业务管理模块错误码
登录与注销	TsdkLoginErrId	登录模块错误码
音视频呼叫	TsdkCallErrId	呼叫模块错误码
IPT业务(点击呼叫)	TsdkCtdErrId	CTD模块错误码
音视频会议	TsdkConfErrId	会控模块错误码
企业通讯录	TsdkEaddrErrId	EADDR模块错误码
即时消息	TsdkImErrId	im模块错误码定义

6.2 日志分析

获取日志

日志路径是应用程序在SDK各组件的初始化阶段自定义路径。

说明

1、因部分组件暂不支持设置路径，默认应用程序的运行目录下，为方便一次性获取日志，建议在初始化时，将SDK各组件的日志指定为相同路径。

2、SDK的日志命名一般以"tup_"或"hme_"为前缀。

日志分析

1. SDK一般格式为：

   [YYYY-MM-DD HH:MM:SS.MS][P:xxxx/T:xxxx][level][filename:line function]log info string

   其中：

   [YYYY-MM-DD HH:MM:SS.MS] 表示日志打印时间，精确到ms；

   [P:xxxx/T:xxxx] 表示日志所在的进程ID和线程ID；

   [level] 表示日志级别，包括错误(Err)、警告(War)、信息(Inf)和调试(Dbg)级别；

   [filename:line function]表示日志所在的文件名，行号和函数；

   log info string 为日志信息描述。

   建议开发者根据时间顺序进行相应的业务调用分析。

2. 对于业务的关键逻辑位置，会打印一条相应的关键信息日志。

```
[2017-05-05 16:29:35.077][P:7920/T:8776][Inf][call][call_msg.c:881 CALLMPROC_MSG_AsynSend]ulMsgID = 0x00000008[CALL_E_EVT_CALL_DESTROY], ulParam123 = [0x7bdfb600, 1, 0], from [P7920] to [P7920]
[2017-05-05 16:29:35.077][P:7920/T:8776][Dbg][call][call_basic.c:969 callbasicDestroyBasicCall]MEDIA_SetAudioPlayDelay(0) 0
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][call][call_basic.c:15740 CallBasicSetVideoMute]callID: 0x7bdfb600, bIsMute=0
[2017-05-05 16:29:35.079][P:7920/T:8776][Inf][call][call_basic.c:15762 CallBasicSetVideoMute]3.
```



```
ulCallID = 2078258688, bIsMute=0
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][call][call_conf.c:5110
CallConfOnEndConfCall]CallConfOnEndConfCall ulConfID=0x0
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][call][call_basic.c:11348
CallBasicGetConfID]CallBasicGetConfID(0x7bdfb600)
[2017-05-05 16:29:35.079][P:7920/T:8776][Err][call][call_basic.c:11354 CallBasicGetConfID]Get Call
ID(0x7bdfb600) Error=0x8002113
[2017-05-05 16:29:35.079][P:7920/T:8776][Err][call][call_conf.c:5123 CallConfOnEndConfCall]con't
find ServerConf by ID=0xffffffff!
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][call][call_main.c:2128 callmainMsgProcModTsp]call
process msg leave, msgId: 0x2904
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][SipTptd][sstpthiglue.c:292
SipStackTptDToTptMsgProc]start process tpt to tptd msg: TPTD_SENDRESULT_SUCC
[2017-05-05 16:29:35.079][P:7920/T:8776][Dbg][SipTptd][sstpthiglue.c:394
SipStackTptDToTptMsgProc]end process tpt to tptd msg: TPTD_SENDRESULT_SUCC
```


7 附录

7.1 Hello World 源码文件

7.1.1 LoginActivity.java

```
/**
 * The type Login activity.
 */
public class LoginActivity extends Activity implements View.OnClickListener {

    private static final String REGISTER_SERVER = "";
    private static final String PORT = "";
    private static final String ACCOUNT = "";
    private static final String PASSWORD = "";
    private String regServer = "218.4.33.69"; // register server address
    private String serverPort = "8443"; // server port
    private String sipNumber = "tup03@huawei.com"; // sip number
    private String password = "Aa123456"; // password
    private EditText regServerEditText;
    private EditText serverPortEditText;
    private EditText usernameEditText;
    private EditText passwordEditText;
    private Button loginButton;
    private IntentFilter filter;

    private BroadcastReceiver receiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if (TupLoginService.TUP_REGISTER_EVENT.equals(action))
            {
                int result = intent.getIntExtra(TupLoginService.REGISTER_RESULT, -1);
                switch (result)
                {
                    case 0:
                        Toast.makeText(LoginActivity.this, getString(R.string.login_success),
Toast.LENGTH_LONG).show();
                        break;
                    default:
                        break;
                }
```

eSDK CloudVC 20.1.RC1 开发指南(Android) 7 附录

```
            }
        }
    };

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

    private void initView()
    {
        regServerEditText = (EditText) findViewById(R.id.et_register_server_address);
        serverPortEditText = (EditText) findViewById(R.id.et_server_port);
        usernameEditText = (EditText) findViewById(R.id.et_username);
        passwordEditText = (EditText) findViewById(R.id.et_password);
        regServerEditText.setText(REGISTER_SERVER);
        usernameEditText.setText(ACCOUNT);
        passwordEditText.setText(PASSWORD);
        serverPortEditText.setText(PORT);
        loginButton = (Button) findViewById(R.id.btn_login);
        loginButton.setOnClickListener(this);
    }

    private void initFilter()
    {
        filter = new IntentFilter();
        filter.addAction(TupLoginService.TUP_REGISTER_EVENT);
        registerReceiver(receiver, filter);
    }

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

    private void login()
    {
        regServer = regServerEditText.getText().toString().trim();
        sipNumber = usernameEditText.getText().toString().trim();
        password = passwordEditText.getText().toString().trim();
        serverPort = serverPortEditText.getText().toString().trim();
        if (TextUtils.isEmpty(sipNumber) || TextUtils.isEmpty(password) || TextUtils.isEmpty(regServer) ||
TextUtils.isEmpty(serverPort))
        {
            Toast.makeText(LoginActivity.this, getString(R.string.account_information_not_empty),
                    Toast.LENGTH_SHORT).show();
            return;
        }
        TupLoginService.getInstance().authorizeLogin(regServer, serverPort, sipNumber, password);
    }
    @Override
    protected void onDestroy()
    {
        unregisterReceiver(receiver);
        super.onDestroy();
    }
}
```


7.1.2 ECApplication.java

```
/**
 * The type EC application.
 */
public class ECApplication extends Application {

    private static Application app;
    private TsdkManager tsdkManager;

    /**
     * Gets app.
     *
     * @return the app
     */
    public static Application getApp()
    {
        return app;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        app = this;
        String appPath = getApplicationInfo().dataDir + "/lib";
        startService(this, appPath);
    }

    /**
     * Init int.
     * @param context the context
     * @param appPath the app path
     * @return the int
     */
    public int startService(Context context, String appPath)
    {
        tsdkManager = TsdkManager.getInstance(context, appPath, ServiceNotify.getInstance());
        TsdkAppInfoParam appInfoParam = new TsdkAppInfoParam();
        appInfoParam.setClientType(TsdkClientType.TSDK_E_CLIENT_MOBILE);
        appInfoParam.setProductName("SoftClient on Mobile");
        appInfoParam.setDeviceSn("123");
        appInfoParam.setSupportAudioAndVideoCall(0);
        appInfoParam.setSupportAudioAndVideoConf(0);
        appInfoParam.setSupportDataConf(0);
        appInfoParam.setSupportCtd(0);
        appInfoParam.setSupportEnterpriseAddressBook(0);
        appInfoParam.setSupportIm(0);
        appInfoParam.setSupportRichMediaMessage(0);

        int ret = tsdkManager.init(appInfoParam);
        return ret;
    }

    @Override
    public void onTerminate() {
        super.onTerminate();
        tsdkManager.uninit();
    }
}
```

7.1.3 TupLoginService.java

```
/**
 * The type login mgr.
 */
public class TupLoginService {

    /**
```



```
 * The constant TUP_REGISTER_EVENT.
 */
public static final String TUP_REGISTER_EVENT = "tup_register_event";

/**
 * The constant REGISTER_RESULT.
 */
public static final String REGISTER_RESULT = "register_result";

private static TupLoginService tupLoginService;

private TupLoginService()
{
}

/**
 * Get instance.
 *
 * @return the instance.
 */
public static synchronized TupLoginService getInstance()
{
    if (tupLoginService == null)
    {
        tupLoginService = new TupLoginService();
    }
    return tupLoginService;
}

/**
 * Authorize login boolean.
 *
 * @param regServer  the reg server
 * @param serverPort the server port
 * @param sipNumber  the sip number
 * @param password   the password
 * @return the boolean
 */
public boolean authorizeLogin(String regServer, String serverPort, String sipNumber, String password)
{
    int ret;

    TsdkLoginParam tsdkLoginParam = new TsdkLoginParam();
    tsdkLoginParam.setUserId(1);
    tsdkLoginParam.setAuthType(TsdkAuthType.TSDK_E_AUTH_NORMAL);
    tsdkLoginParam.setUserName(sipNumber);
    tsdkLoginParam.setPassword(password);
    tsdkLoginParam.setServerAddr(regServer);
    tsdkLoginParam.setServerPort(Integer.parseInt(serverPort));
    tsdkLoginParam.setServerVersion("");
    tsdkLoginParam.setServerType(TsdkServerType.TSDK_E_SERVER_TYPE_PORTAL);
    tsdkLoginParam.setUserTiket("");

    ret = TsdkManager.getInstance().getLoginManager().login(tsdkLoginParam);
    if (ret != 0) {
        return false;
    }
    return true;
}

/**
 * Login un init int.
 *
 * @return the int
 */
public int logout() {
    int ret = TsdkManager.getInstance().getLoginManager().logout();
    if (ret != 0) {
        Log.e("logout", "login is failed, return " + ret);
```

```
        }
        return ret;
    }

    public void handleAuthSuccess() {
        Log.d("TupLoginService", "onEvtAuthSuccess.");

        Intent intent = new Intent();
        intent.setAction(TUP_REGISTER_EVENT);
        intent.putExtra(REGISTER_RESULT, 0);
        ECApplication.getApp().sendBroadcast(intent);
    }
}
```

7.1.4 ServiceNotify.java

```
/**
 * The type service notify.
 */
public class ServiceNotify implements TsdkNotify {

    private static ServiceNotify instance;

    public static ServiceNotify getInstance() {
        if (null == instance) {
            instance = new ServiceNotify();
        }
        return instance;
    }

    @Override
    public void onEvtAuthSuccess(int i, TsdkImLoginParam tsdkImLoginParam) {
        TupLoginService.getInstance().handleAuthSuccess();
    }

    @Override
    public void onEvtAuthFailed(int i, TsdkCommonResult tsdkCommonResult) {

    }

    @Override
    public void onEvtAuthRefreshFailed(int i, TsdkCommonResult tsdkCommonResult) {

    }

    @Override
    public void onEvtLoginSuccess(int i) {

    }

    @Override
    public void onEvtLoginFailed(int i, TsdkCommonResult tsdkCommonResult) {

    }

    @Override
    public void onEvtLogoutSuccess(int i) {

    }

    @Override
    public void onEvtLogoutFailed(int i, TsdkCommonResult tsdkCommonResult) {

    }

    @Override
    public void onEvtForceLogout(int i) {

    }
```

eSDK CloudVC 20.1.RC1 开发指南(Android)


```
@Override
public void onEvtVoipAccountStatus(int i, TsdkVoipAccountInfo tsdkVoipAccountInfo) {

}

@Override
public void onEvtCallStartResult(TsdkCall tsdkCall, TsdkCommonResult tsdkCommonResult) {

}

@Override
public void onEvtCallIncoming(TsdkCall tsdkCall, Boolean aBoolean) {

}

@Override
public void onEvtCallOutgoing(TsdkCall tsdkCall) {

}

@Override
public void onEvtCallRingback(TsdkCall tsdkCall) {

}

@Override
public void onEvtCallRtpCreated(TsdkCall tsdkCall) {

}

@Override
public void onEvtCallConnected(TsdkCall tsdkCall) {

}

@Override
public void onEvtCallEnded(TsdkCall tsdkCall) {

}

@Override
public void onEvtCallDestroy(TsdkCall tsdkCall) {

}

@Override
public void onEvtOpenVideoReq(TsdkCall tsdkCall, TsdkVideoOrientation tsdkVideoOrientation) {

}

@Override
public void onEvtRefuseOpenVideoInd(TsdkCall tsdkCall) {

}

@Override
public void onEvtCloseVideoInd(TsdkCall tsdkCall) {

}

@Override
public void onEvtOpenVideoInd(TsdkCall tsdkCall) {

}

@Override
public void onEvtRefreshViewInd(TsdkCall tsdkCall, TsdkVideoViewRefresh tsdkVideoViewRefresh) {
```

```
}

@Override
public void onEvtCallRouteChange(TsdkCall tsdkCall, int i) {

}

@Override
public void onEvtPlayMediaEnd(int i) {

}

@Override
public void onEvtSessionModified(TsdkCall tsdkCall, TsdkSessionModified tsdkSessionModified) {

}

@Override
public void onEvtSessionCodec(TsdkCall tsdkCall, TsdkSessionCodec tsdkSessionCodec) {

}

@Override
public void onEvtHoldSuccess(TsdkCall tsdkCall) {

}

@Override
public void onEvtHoldFailed(TsdkCall tsdkCall) {

}

@Override
public void onEvtUnholdSuccess(TsdkCall tsdkCall) {

}

@Override
public void onEvtUnholdFailed(TsdkCall tsdkCall) {

}

@Override
public void onEvtEndcallFailed(TsdkCall tsdkCall, TsdkCommonResult tsdkCommonResult) {

}

@Override
public void onEvtDivertFailed(TsdkCall tsdkCall) {

}

@Override
public void onEvtBldTransferSuccess(TsdkCall tsdkCall) {

}

@Override
public void onEvtBldTransferFailed(TsdkCall tsdkCall) {

}

@Override
public void onEvtSetIptServiceResult(int i, TsdkSetIptServiceResult tsdkSetIptServiceResult) {

}

@Override
public void onEvtIptServiceInfo(TsdkIptServiceInfoSet tsdkIptServiceInfoSet) {
```

```
    }

    @Override
    public void onEvtBookConfResult(TsdkCommonResult tsdkCommonResult, TsdkConfBaseInfo
tsdkConfBaseInfo) {

    }

    @Override
    public void onEvtQueryConfListResult(TsdkCommonResult tsdkCommonResult, TsdkConfListInfo
tsdkConfListInfo) {

    }

    @Override
    public void onEvtQueryConfDetailResult(TsdkCommonResult tsdkCommonResult, TsdkConfDetailInfo
tsdkConfDetailInfo) {

    }

    @Override
    public void onEvtJoinConfResult(TsdkConference tsdkConference, TsdkCommonResult
tsdkCommonResult, TsdkJoinConfIndInfo tsdkJoinConfIndInfo) {

    }

    @Override
    public void onEvtGetDataconfParamResult(TsdkConference tsdkConference, TsdkCommonResult
tsdkCommonResult) {

    }

    @Override
    public void onEvtConfctrlOperationResult(TsdkConference tsdkConference, TsdkConfOperationResult
tsdkConfOperationResult) {

    }

    @Override
    public void onEvtInfoAndStatusUpdate(TsdkConference tsdkConference) {

    }

    @Override
    public void onEvtSpeakerInd(TsdkConference tsdkConference, TsdkConfSpeakerInfo tsdkConfSpeakerInfo)
{

    }

    @Override
    public void onEvtRequestConfRightFailed(TsdkConference tsdkConference, TsdkCommonResult
tsdkCommonResult) {

    }

    @Override
    public void onEvtConfIncomingInd(TsdkConference tsdkConference) {

    }

    @Override
    public void onEvtConfEndInd(TsdkConference tsdkConference) {

    }

    @Override
    public void onEvtJoinDataConfResult(TsdkConference tsdkConference, TsdkCommonResult
tsdkCommonResult) {
```

eSDK CloudVC 20.1.RC1 开发指南(Android) 7 附录

```
    }

    @Override
    public void onEvtCtdStartCallResult(int i, TsdkCommonResult tsdkCommonResult) {

    }

    @Override
    public void onEvtCtdEndCallResult(int i, TsdkCommonResult tsdkCommonResult) {

    }

    @Override
    public void onEvtCtdCallStatusNotify(int i, TsdkCtdCallStatus tsdkCtdCallStatus) {

    }

    @Override
    public void onEvtSearchContactsResult(int i, TsdkCommonResult tsdkCommonResult,
TsdkSearchContactsResult tsdkSearchContactsResult) {

    }

    @Override
    public void onEvtSearchDeptResult(int i, TsdkCommonResult tsdkCommonResult,
TsdkSearchDepartmentResult tsdkSearchDepartmentResult) {

    }

    @Override
    public void onEvtGetIconResult(int i, TsdkCommonResult tsdkCommonResult, TsdkGetIconResult
tsdkGetIconResult) {

    }
}
```

7.1.5 activity_login.xml

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

    <RelativeLayout
        android:layout_width="300dp"
        android:layout_height="wrap_content">

        <ScrollView
            android:layout_width="match_parent"
            android:layout_height="wrap_content" >

            <LinearLayout
                android:id="@+id/llTextContainer"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical">

                <EditText
                    android:id="@+id/et_register_server_address"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_centerVertical="true"
                    android:hint="@string/register_server"
                    android:inputType="textVisiblePassword"
                    android:singleLine="true"
                    android:textSize="13sp" />
```



```
            <EditText
                android:id="@+id/et_server_port"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_centerVertical="true"
                android:hint="@string/server_port"
                android:inputType="textVisiblePassword"
                android:singleLine="true"
                android:textSize="13sp" />
            <EditText
                android:id="@+id/et_username"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_centerVertical="true"
                android:hint="@string/account"
                android:inputType="textVisiblePassword"
                android:singleLine="true"
                android:textSize="13sp" />
            <EditText
                android:id="@+id/et_password"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_centerVertical="true"
                android:hint="@string/password"
                android:inputType="textPassword"
                android:singleLine="true"
                android:textSize="13sp" >
            </EditText>


            <Button
                android:id="@+id/btn_login"
                android:layout_width="match_parent"
                android:layout_height="43dp"
                android:layout_marginTop="10dp"
                android:text="@string/login"
                android:textSize="15sp" />

        </LinearLayout>
    </ScrollView>
</RelativeLayout>


</LinearLayout>
```

7.1.6 AndroidMainifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.huawei.esdk.tup.helloworld">

    <!-- uses permission -->
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>


    <application
        android:name=".application.ECApplication"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
```

eSDK CloudVC 20.1.RC1 开发指南(Android)


```
        android:supportsRtl="true">
        <activity android:name=".ui.LoginActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

7.1.7 Strings

```
<resources>
    <string name="app_name">HelloWrold</string>
    <string name="register_server">RegServer</string>
    <string name="server_port">ServerPort</string>
    <string name="account">Account</string>
    <string name="password">Password</string>
    <string name="login">Login</string>
    <string name="login_success">Login success</string>
    <string name="account_information_not_empty">Please enter your login parameters</string>
</resources>
```

7.1.8 build.gradle

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

android {
    compileSdkVersion 25
    buildToolsVersion "26.0.1"

    defaultConfig {
        jackOptions {
            enabled true
        }
    }

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

    defaultConfig {
        applicationId "com.huawei.esdk.tup.helloworld"
        minSdkVersion 15
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    sourceSets {
        main {
            assets.srcDirs = ['assets', 'src/main/assets', 'src/main/assets/']
            jniLibs.srcDirs = ['libs']
        }
        instrumentTest.setRoot('tests')
        debug.setRoot('build-types/debug')
        release.setRoot('build-types/release')
    }
}
```

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    testCompile 'junit:junit:4.12'
    compile 'com.google.code.gson:gson:2.3.1'
}
```

7.2 全平台操作系统要求

eSDK CloudVC支持的OS

操作平台	设备类型
windows	Windows 7(32位及64位)/Windows 8(32位及64位)/Windows 10(32位及64位)操作系统,X86硬件
Mac	OSX10.7/10.8/10.9/10.10/10.11/10.12，32位与64位操作系统,X86硬件
Android Phone	Android 5.0~Android 7.0 基于ARMv7 Neon芯片及以上架构的CPU固件版本
Android Pad	Android 5.0~Android7.0 基于ARMv7 Neon芯片及以上架构的CPU固件版本
iPhone	iPhone7 Plus、iPhone7、iPhone6 Plus、iPhone6、iPhone SE、iPhone5S，iOS 9到iOS 11 固件版本
iPad	iPad Pro、iPad Air2、iPad Air、iPad mini、iPad mini2、iPad mini3、iPad2、iPad3、iPad4，iOS 9到iOS 11 固件版本

eSDK CloudVC 20.1.RC1 开发指南(Android)

8 修订记录

发布日期	文档版本	修订说明
2019-03-15	05	配套19.1.RC1版本，文档刷新发布，主要更新包括： 1、新增即时消息功能； 2、新增会议录制功能； 3、优化登录逻辑，变更相关回调事件
2019-01-02	04	配套19.0.0版本，文档刷新发布。 刷新全平台操作系统要求。
2018-10-12	03	配套19.0.RC1版本，文档刷新发布，主要更新包括： 1、新增融合会议Hosted组网下支持匿名会议； 2、优化会议控制能力，提升会议控制实时性，提升会议用户体验。
2018-07-07	02	配套6.1.0版本，文档刷新发布，主要更新包括： 1、SDK新增对融合会议On-premise组网支持； 2、SDK新增会议中发送消息功能；
2018-05-10	01	配套6.1.RC1版本(暂仅支持融合会议Hosted组网)，文档首次发布。