eSDK CloudVC 20.1.RC1 开发指南(iOS)

eSDK CloudVC 20.1.RC1 开发指南 (iOS)

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

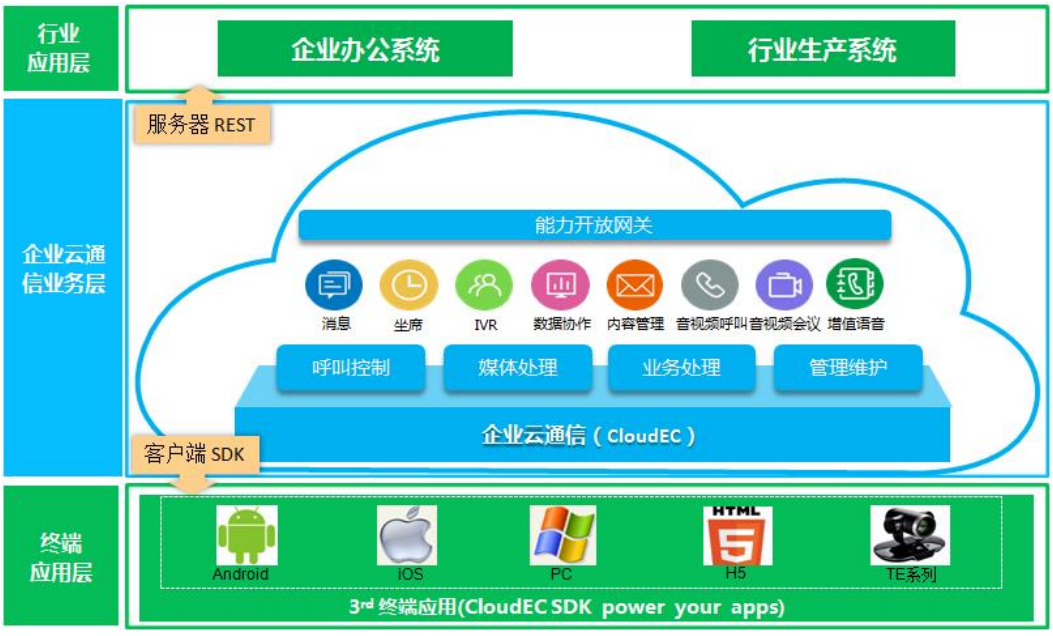
目 录

eSDK CloudVC 20.1.RC1 开发指南(iOS)

1 eSDK CloudVC 简介

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?

CloudVC SDK二次开发目前提供Windows动态库调用方式，将华为eSDK CloudEC二次开发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Windows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**。

2 内容导航

须知

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

本开发指南主要描述CloudVC能力开放的常用功能，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。包括如何通过社区网站获取资料，Sample codes下载链接等。
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CloudVC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：
   - **登录与注销**
   - **音视频呼叫**
   - **会议**
4. **问题定位**：介绍开发过程中常见问题的定位方法。
5. **开发指南修订记录**：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**。

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 EC**”，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 EC**”，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V100R001C00版本即可）

具体所需资源如**表1 资源说明**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。
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Xcode 8.0**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 Codes 列表

Sample Code名称	描述
CloudEC_Client_API_Demo_iOS	CloudVC客户端SDK全量功能Demo，具备音视频呼叫、音视频设备管理会议管理、会议接入、会议控制、桌面协同与共享等基本功能。 暂仅适用于CloudVC解决方案下的纯会议组网。

3.3 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：

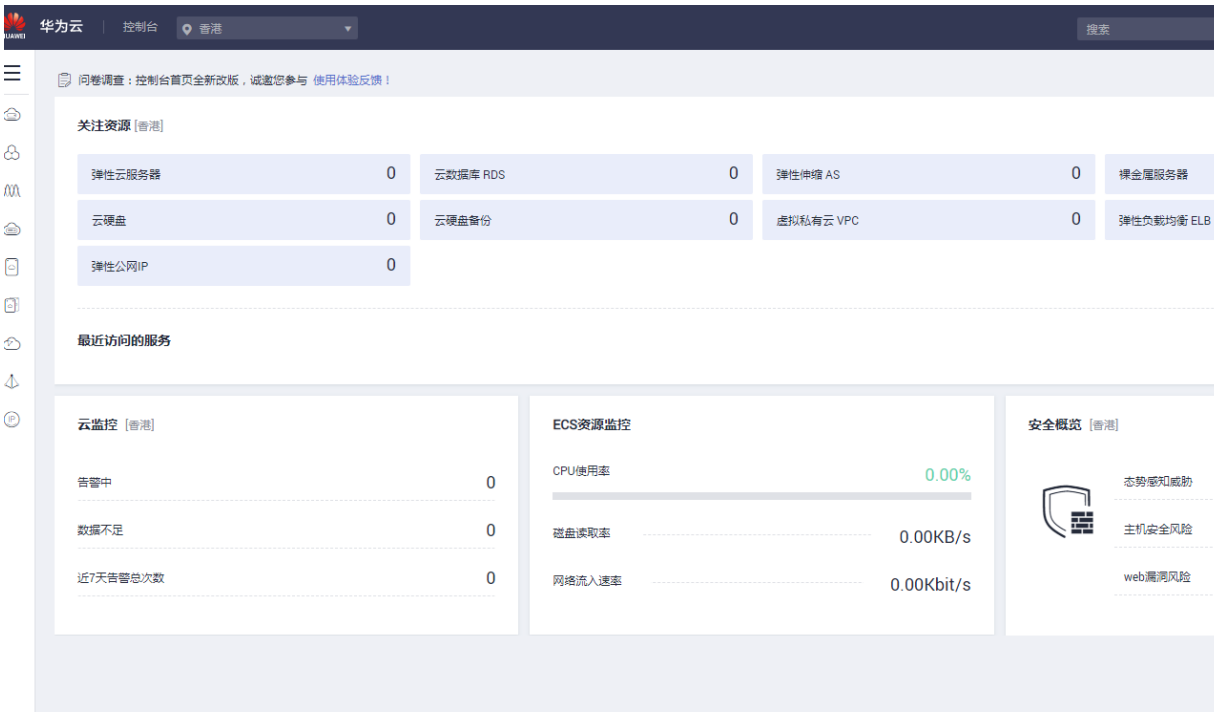
步骤1 登录**华为云开发者社区**。

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。

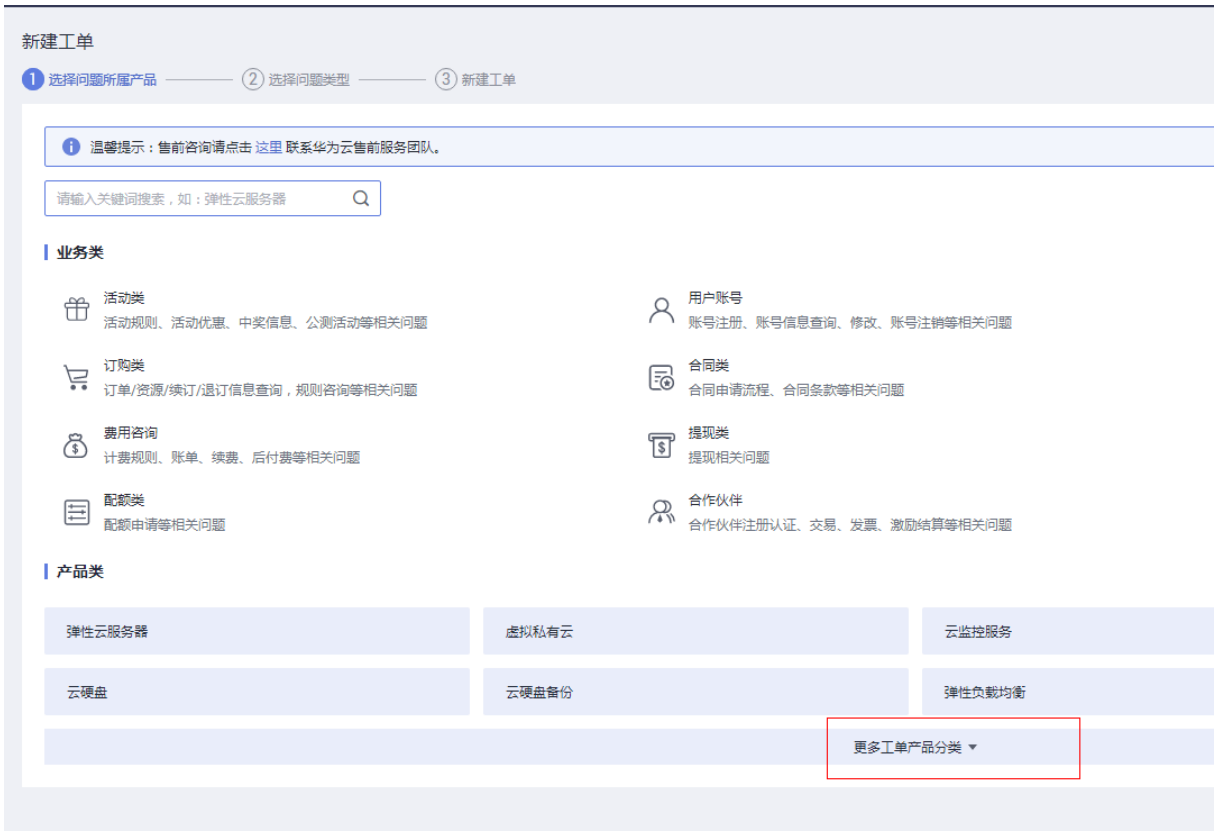
步骤2 单击“**控制台**”，进入控制台页面。

步骤3 在控制台中点击**工单，新建工单**

eSDK CloudVC 20.1.RC1 开发指南(iOS)



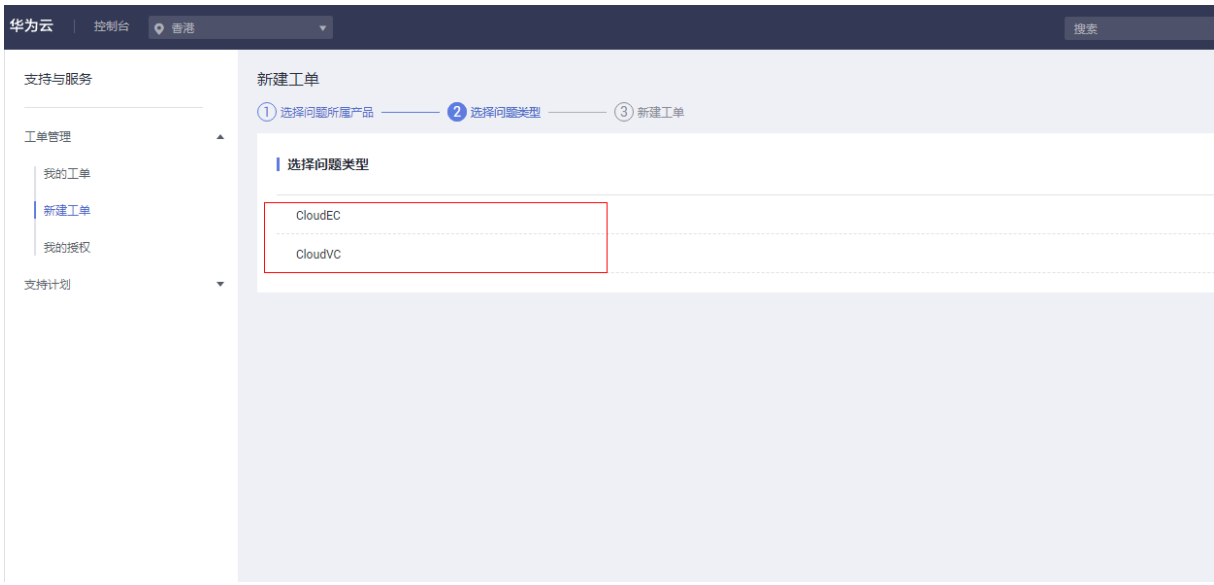
步骤4 在新建工单中，点击**更多工单产品分类**。



在ICT开发者业务中选择**企业通信**

eSDK CloudVC 20.1.RC1 开发指南(iOS)

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

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

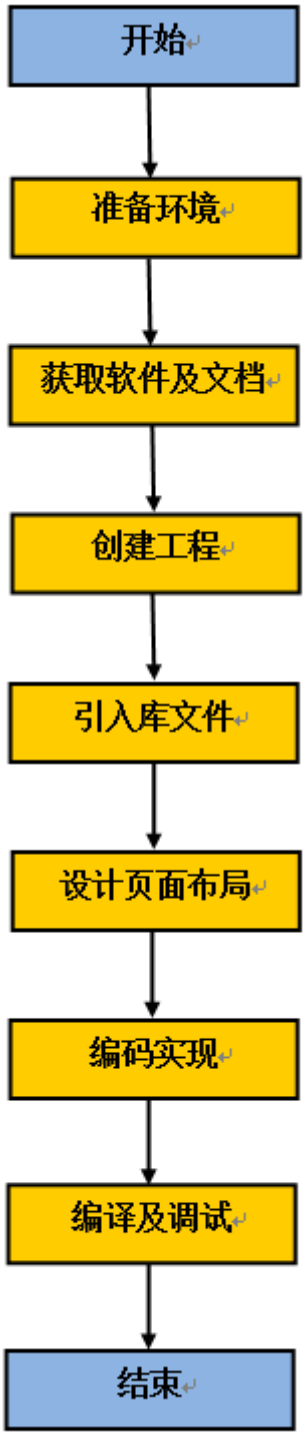
4.1 概述

本示例以C语言调用SDK接口，完成简单的登录鉴权功能为例，向您展示如何使用iPhone平台下的eSDK CloudVC iOS快速进行二次集成开发。

开发过程中的故障处理请参考**问题定位**。

快速入门流程如下：

eSDK CloudVC 20.1.RC1 开发指南(iOS)



4.2 准备环境

开发工具

- 操作系统： iOS 8.0及以上版本。
- Xcode：Xcode 8及以上。

4.3 获取软件及文档

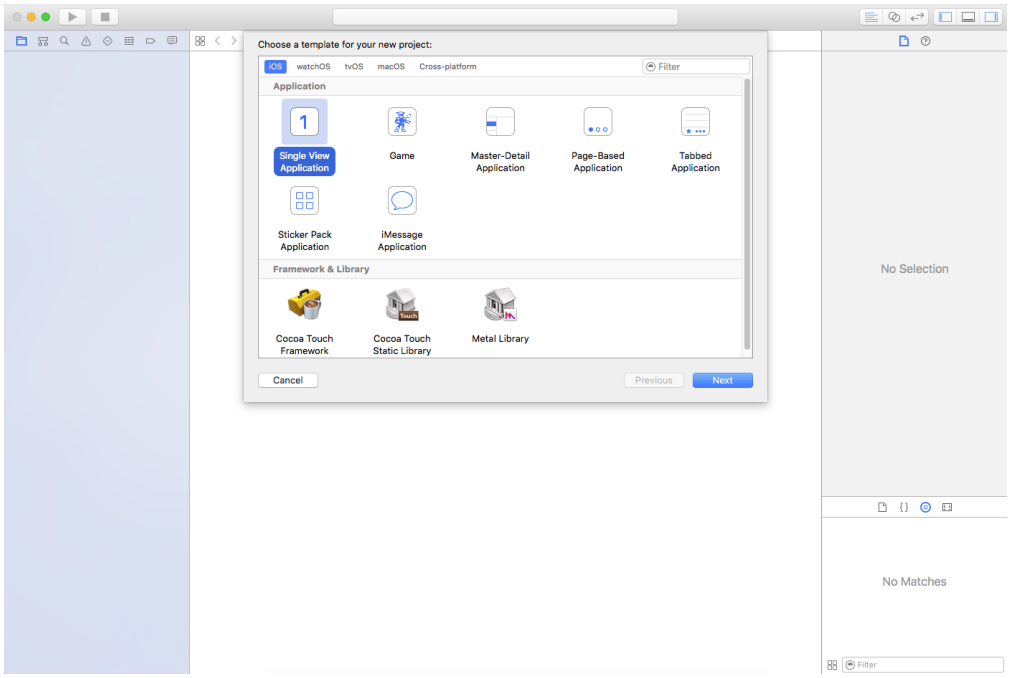
SDK软件包名称：eSDK_VC_API_VX.X.X_iOS.zip

文档名称：

SDK和文档下载路径：参见**SDK下载路径**。

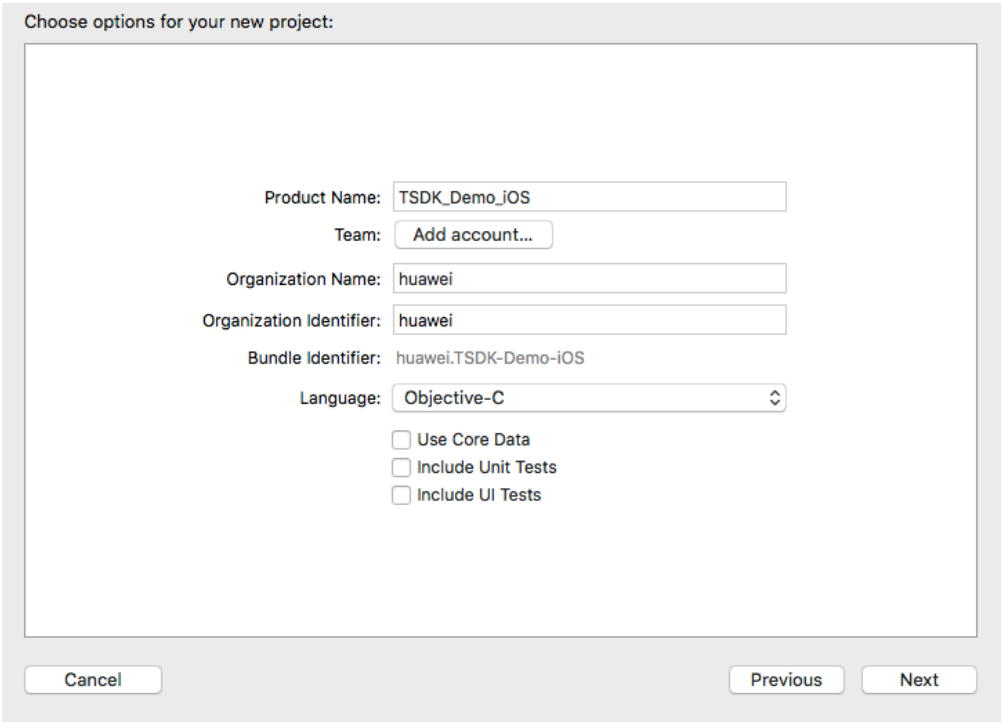
4.4 创建工程

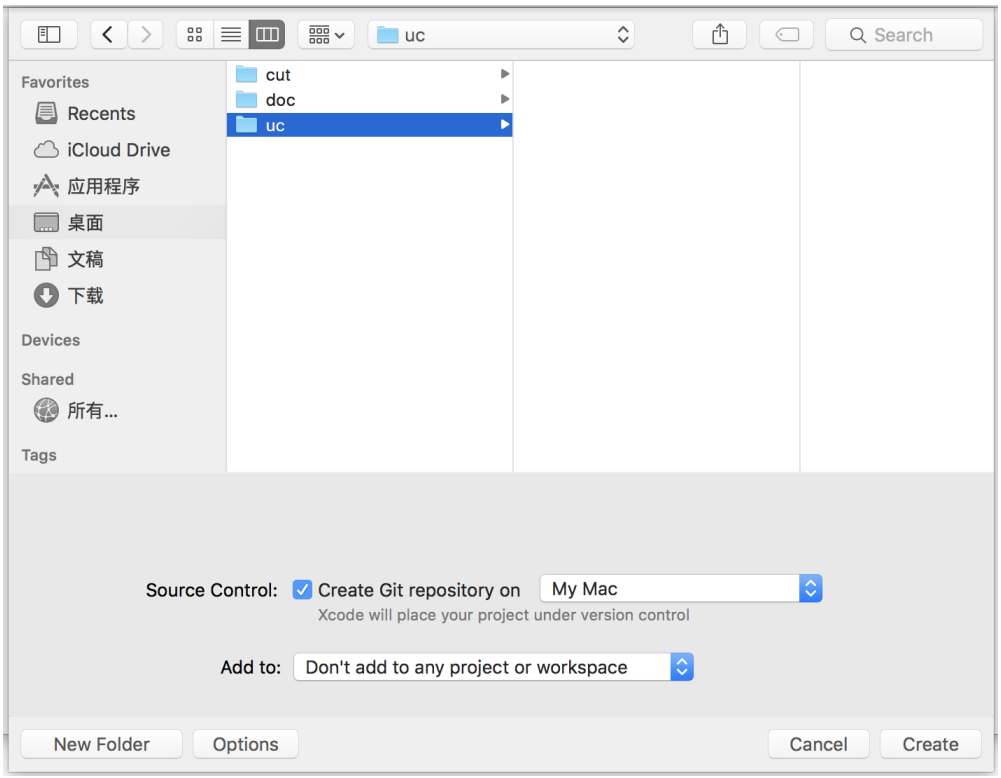
步骤1 打开Xcode，选择“File > New > Project”。系统显示选择模板界面；选择工程类型“Single View Application”，点击“Next”。系统显示配置界面。



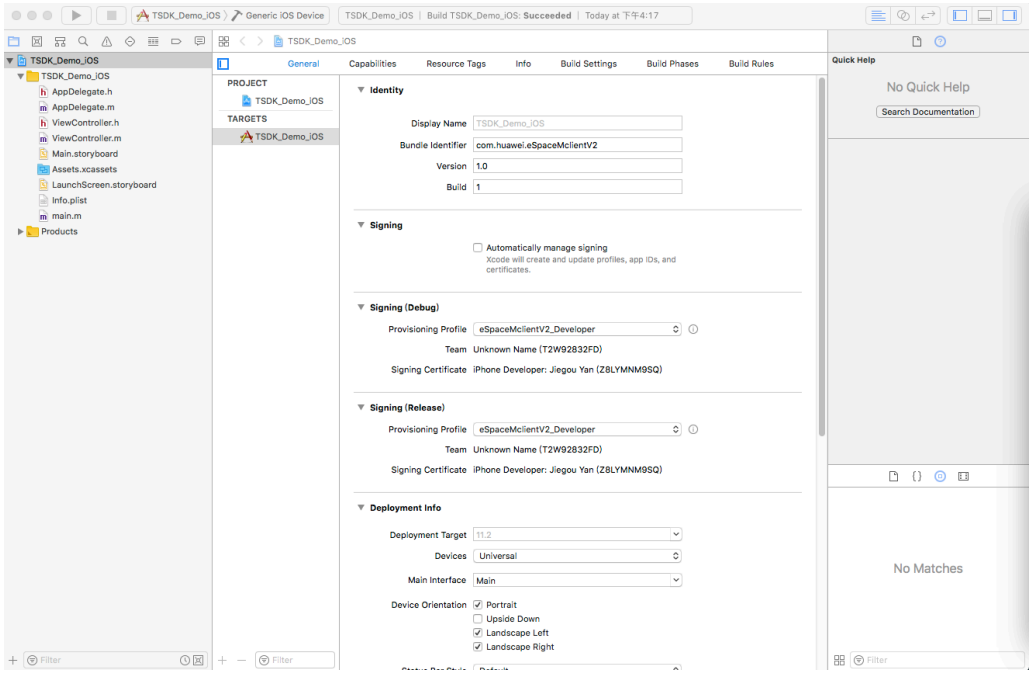
步骤2 在“Product Name” 处输入“TSDK_Demo_iOS”，点击“Next”。选择一个工程路径。

eSDK CloudVC 20.1.RC1 开发指南(iOS)





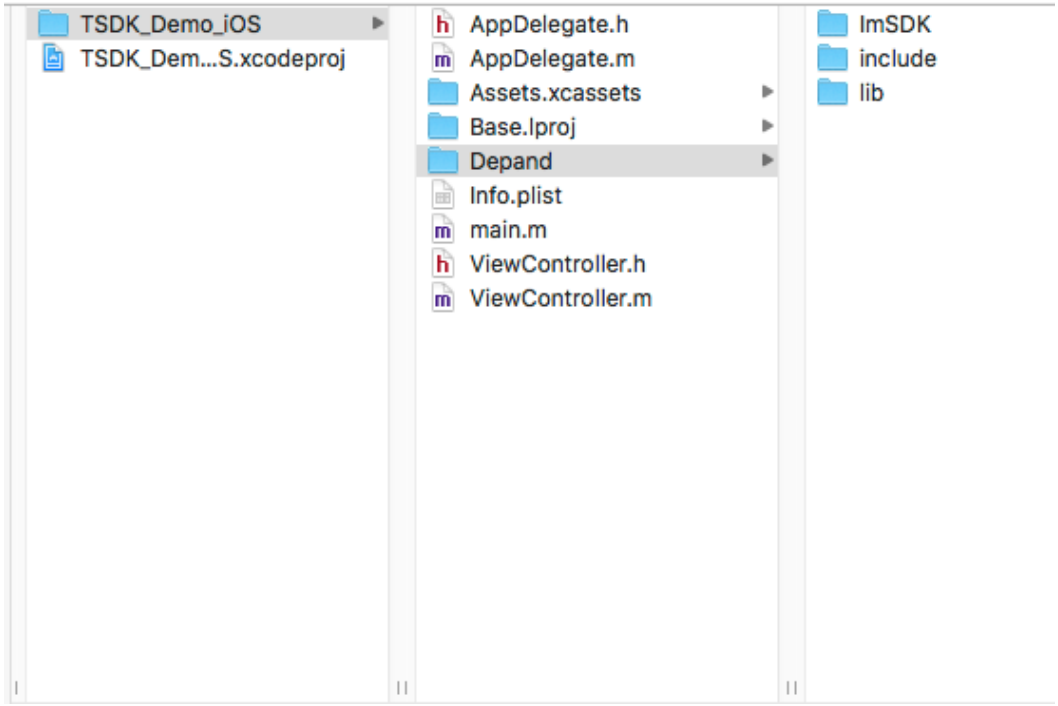
步骤3 单击选择工程存放目录路径，点击“Create”，系统显示工程主界面。



----结束

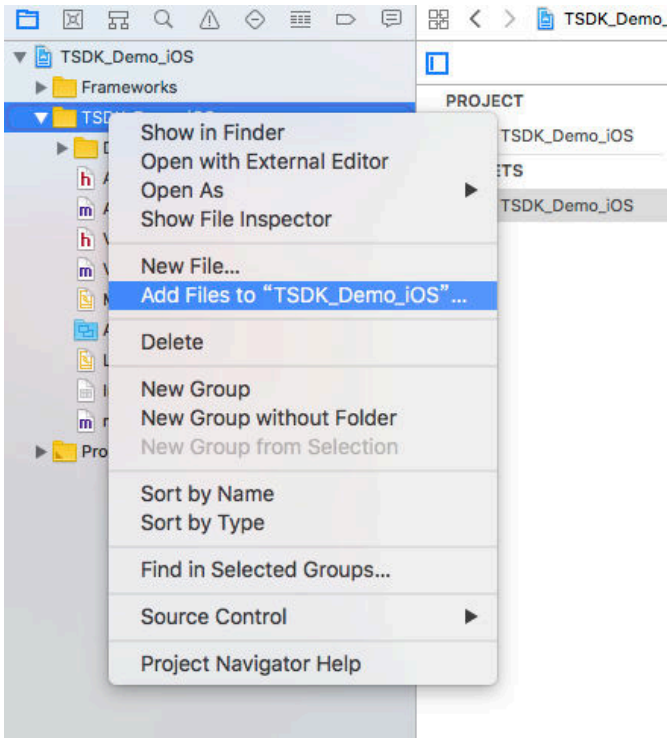
4.5 引入库文件

步骤1 下载工程所需的库文件包，解压缩。在工程里新建文件夹，命名为“Depend”，把库文件lib以及头文件include放置在Depend文件夹里面。

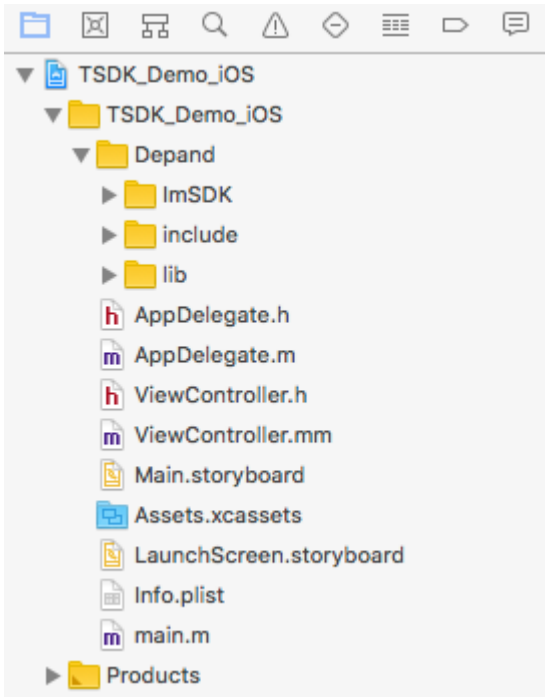


步骤2 右键单击工程左侧“eSDK_TUP_Demo_iOS”项目名称，右键选择“Add Files to “eSDK_TUP_Demo_iOS”...”，系统显示库文件添加页面。

eSDK CloudVC 20.1.RC1 开发指南(iOS)



步骤3 根据库文件存放目录路径（步骤1），将” Depend”文件夹添加到工程中，点击“Add”，完成库文件的添加。



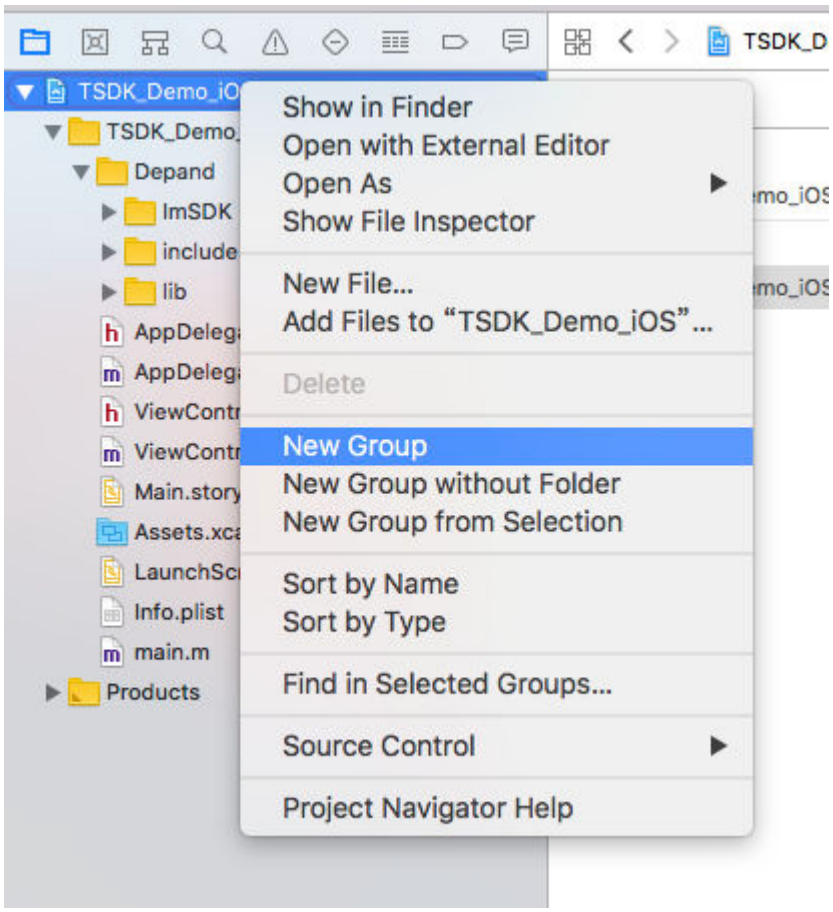
----结束

4.6 添加依赖库

须知

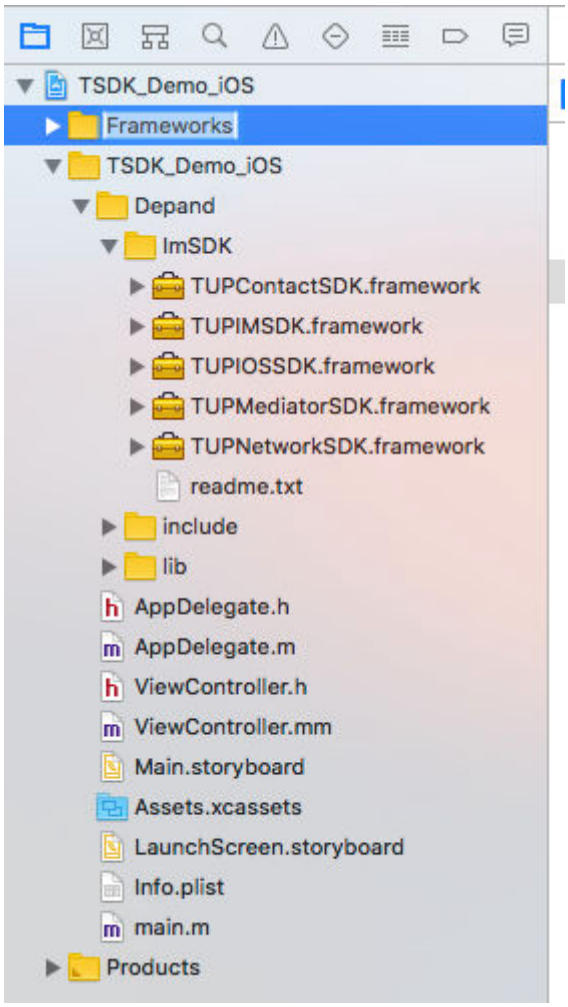
由于TUP的库文件是静态库，因此依赖库需要手动添加，下面只介绍libc++.tbd的添加，其他库添加方式相同。

步骤1 选中工程，右键，选择“New Group”。

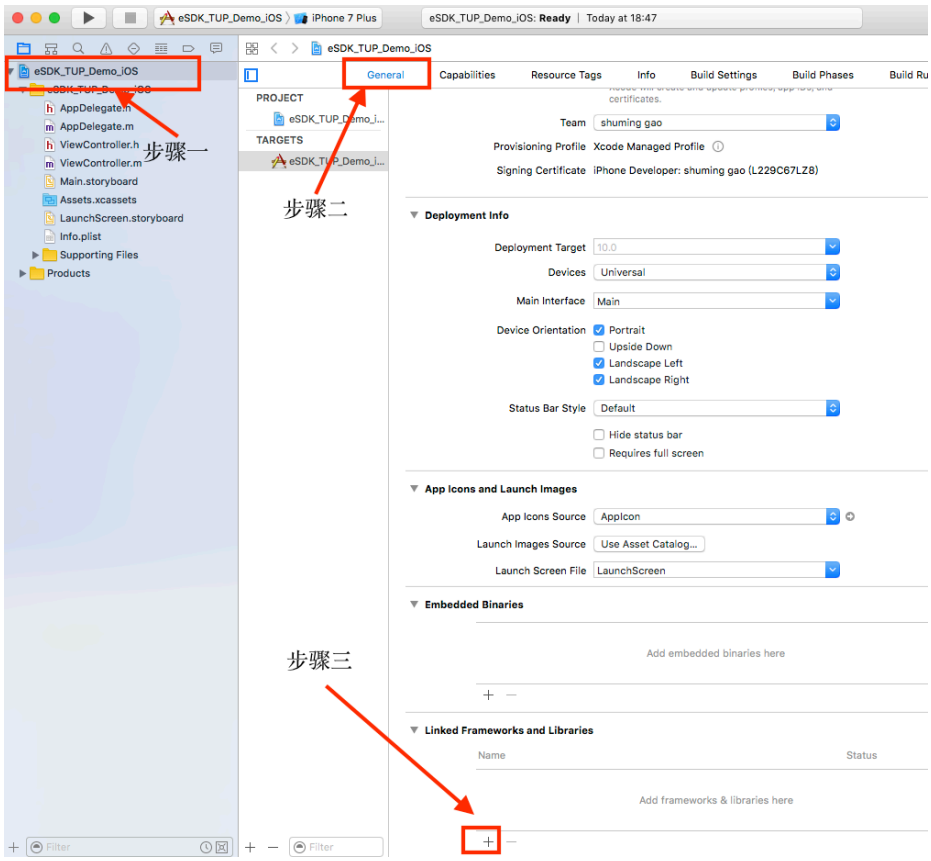


步骤2 修改group的名字为“Frameworks”。

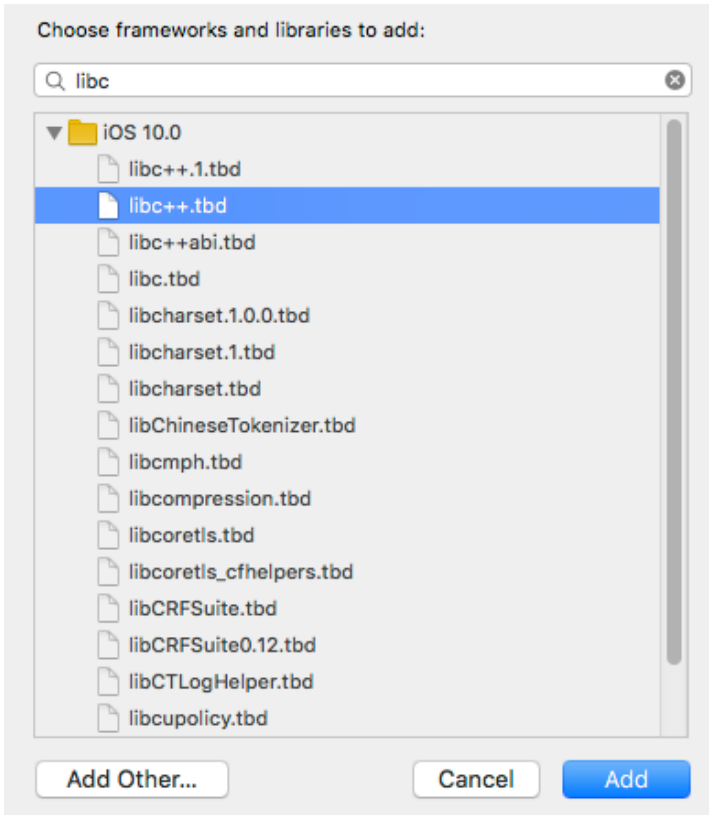
eSDK CloudVC 20.1.RC1 开发指南(iOS)



步骤3 按照下图依次在点击步骤1~3，弹出添加依赖库文件的对话框。

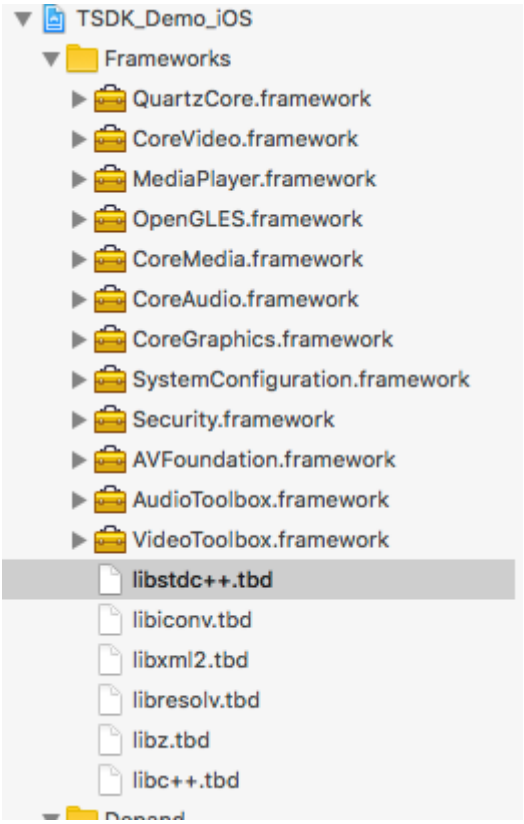


步骤4 在搜索框中查找libc++.tbd，选中，单击“Add”。



eSDK CloudVC 20.1.RC1 开发指南(iOS)

步骤5 将下图所示的依赖库按照上诉方法添加至工程。



说明

Foundation.framework :包含Cocoa Foundation层的类和方法。

UIKit.framework :包含iOS应用程序用户界面层使用的类和方法。

OpenGLES.framework： 包含OpenGL ES接口。OpenGL ES框架是OpenGL跨平台2D和3D渲染库的跨平台版本。

AudioToolbox.framework :包含处理音频流数据以及播放或录制音频的接口。

VedioToolbox.framework:提供硬解码和硬编码API

AVFoundation.framework :包含播放或录制音频的Objective-C接口。

CoreAudio.framework:包含Core Audio框架使用的各种数据类型。

CoreData.framework :包含管理应用程序数据模型的接口。

MediaPlayer.framework： 包含显示全屏视频的接口。

SystemConfiguration.framework:包含用于处理设备网络配置的接口。

Security.framework :包含管理证书、公钥私钥以及信任策略的接口。

QuartzCore.framework :包含Core Animation接口。

CoreVideo.framework :包含操作音频和视频的底层例程。

CoreMedia.framework :包含操作音频和视频的底层例程。

AVFoundation.framework :包含播放或录制音频的Objective-C接口。

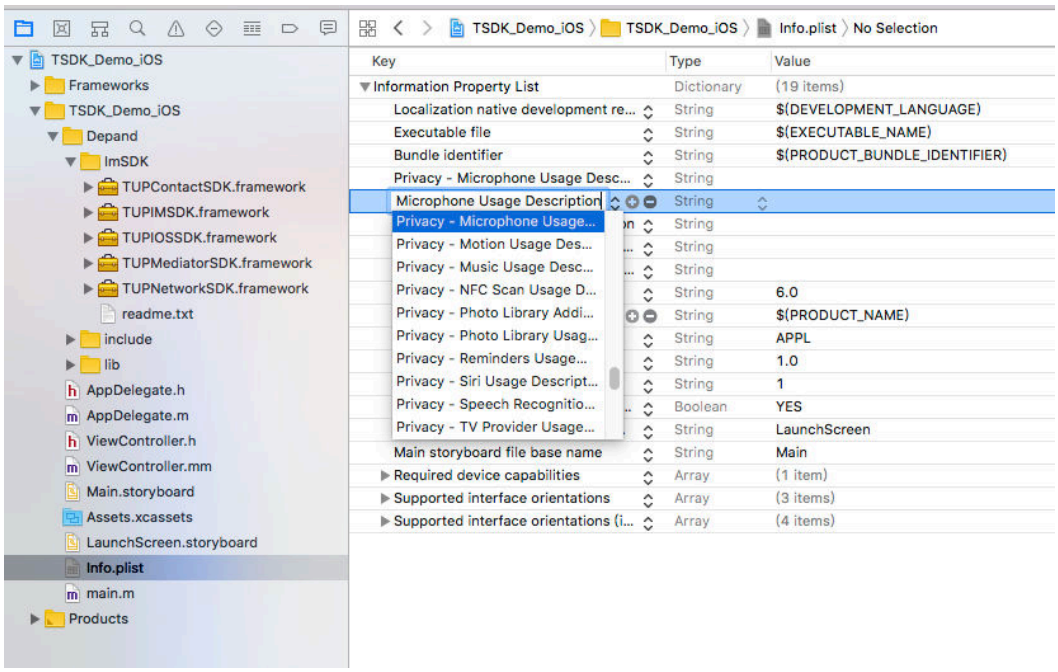
MobileCoreServices.framework :定义系统支持的统一类型标识符（UTIs）.

----结束

4.7 工程配置

步骤1 iOS10后，我们需要在工程主界面Navigator的Info.plist文件中添加权限字段。选择Info.plist文件，右击任意一行，选中“Add Row”，直接输入名称” Privacy - Microphone Usage Description ”，按回车键完成添加。

按照此方法依次添加“Privacy - Camera Usage Description”、“Privacy - Contacts Usage Description”和“Privacy - Photo Library Usage Description”。



说明

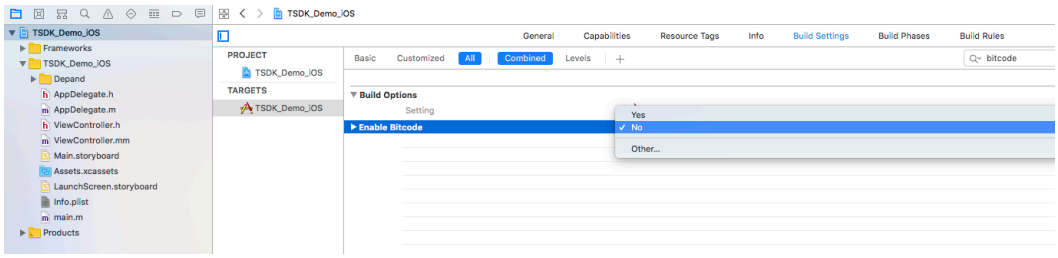
麦克风权限：Privacy - Microphone Usage Description

相机权限： Privacy - Camera Usage Description

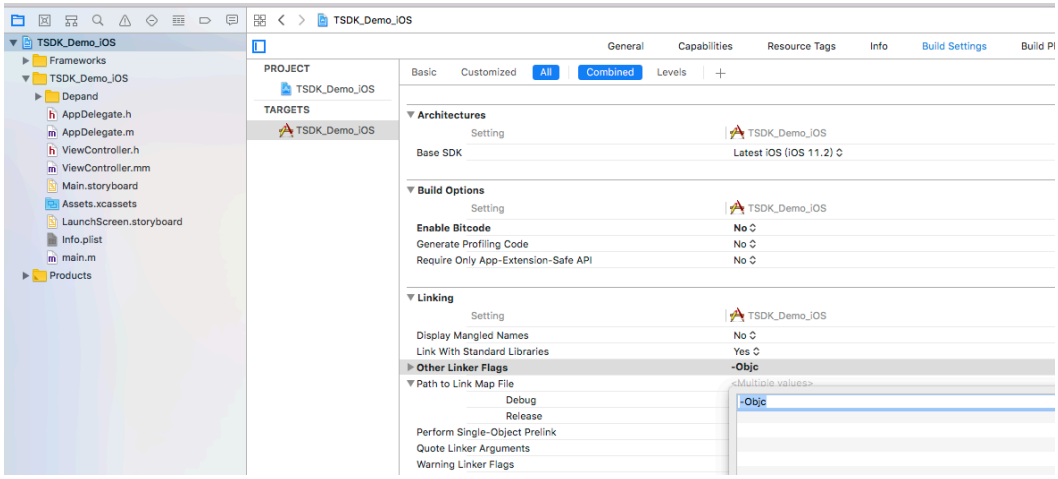
相册权限： Privacy - Photo Library Usage Description

通讯录权限：Privacy - Contacts Usage Description

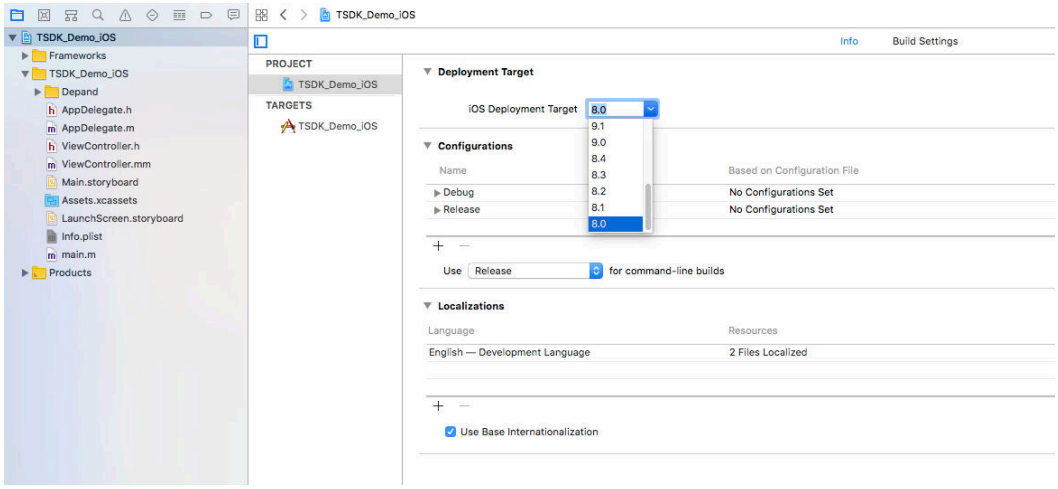
步骤2 点击Navigator下的工程名称，选择“Build Settings>Build Option>Enable Bitcode>NO”。



步骤3 点击Navigator下的工程名称，选择“Build Settings>Linking>Other Linker Flags”，添加“-ObjC”。

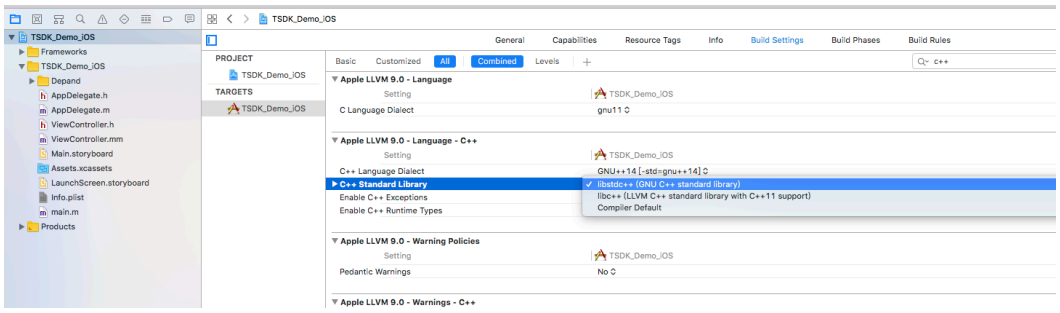


步骤4 真机调试前，需要根据手机系统版本修改Deployment Target，需使其小于等于手机系统版本。

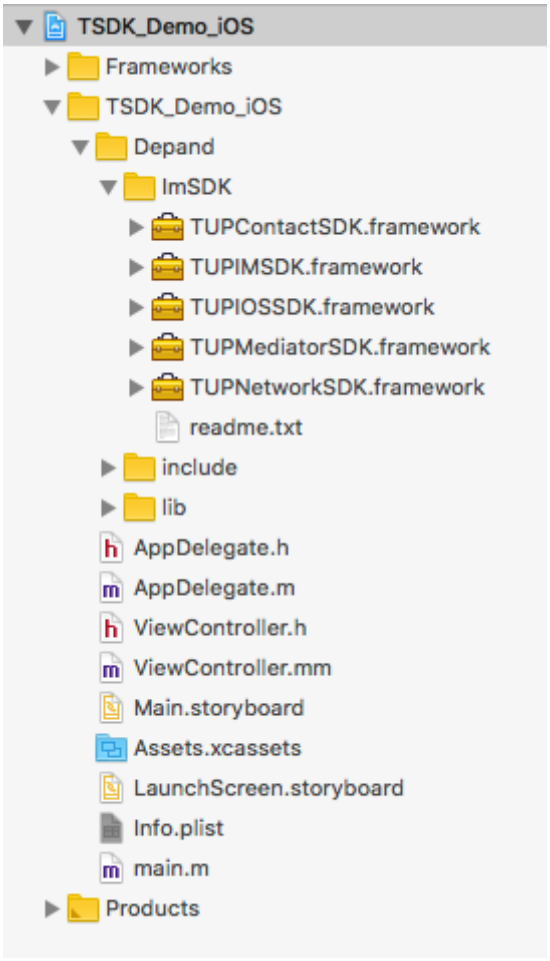


步骤5 TUP库采用了C++的混编方式码，您的工程还需要进行其他配置。

1. 点击“TARGETS>Build Settings>Apple LLMV7.0-language-C++”，将“C++ Standard Library”的值设为“libstdc++（GUN C++ standrary library）”。



2. 将工程目录中的某一个实现过文件后缀手动改为“.mm”。



----结束

4.8 编码实现

源码链接：**ViewController.mm**.

代码参考

点击上述结构中的文件名获取文件源码，并向各文件填充代码。部分代码片段参考如下：

- 选择ViewController.mm，完成TUP库Login组件初始化。

```
TSDK_VOID onTSDKNotifications(TSDK_UINT32 msgid, TSDK_UINT32 param1, TSDK_UINT32 param2,
TSDK_VOID *data)
{
    NSLog(@"onTUPLoginNotifications : %#x",msgid);
    dispatch_async(dispatch_get_main_queue(), ^{
        switch (msgid)
        {
            case TSDK_E_LOGIN_EVT_AUTH_SUCCESS:
            {
                NSLog(@"Uportal login success !");
                [ViewController showMessages:@"Uportal login success"];
            }
                break;
            case TSDK_E_LOGIN_EVT_AUTH_FAILED:
            {
```

eSDK CloudVC 20.1.RC1 开发指南(iOS)


```
                NSLog(@"Uportal login fail !");
                [ViewController showMessages:@"Uportal login fail"];
            }
                break;
            default:
                break;
        }
    });
}
-(BOOL)initUportalLoginService
{
    TSDK_S_LOG_PARAM logParam;
    memset(&logParam, 0, sizeof(TSDK_S_LOG_PARAM));
    NSString *logPath = [[NSHomeDirectory() stringByAppendingPathComponent:@"Documents"]
stringByAppendingString:@"/TUPC60log"];
    NSString *path = [logPath stringByAppendingString:@"/tsdk"];
    logParam.level = TSDK_E_LOG_DEBUG;
    logParam.file_count = 1;
    logParam.max_size_kb = 4*1024;
    strcpy(logParam.path, [path UTF8String]);
    TSDK_RESULT result = tsdk_set_config_param(TSDK_E_CONFIG_LOG_PARAM, &logParam);

    TSDK_S_APP_INFO_PARAM app_info;
    memset(&app_info, 0, sizeof(TSDK_S_APP_INFO_PARAM));
    app_info.client_type = TSDK_E_CLIENT_MOBILE;
    strcpy(app_info.product_name, "SoftClient on Mobile");
    app_info.support_audio_and_video_call = TSDK_TRUE;
    app_info.support_ctd = TSDK_TRUE;
    app_info.support_audio_and_video_conf = TSDK_TRUE;
    app_info.support_enterprise_address_book = TSDK_TRUE;
    result = tsdk_init(&app_info ,&onTSDKNotifications);
    return result == TSDK_SUCCESS ? YES : NO;

}
```

- 发送登录请求

```
NSString *account = @"Account";
NSString *password = @"Password";
NSString *serverAddress = @"192.168.1.100";
int port = 8443;

-(BOOL)loginAuthorizeWithServerAddress:(NSString *)serverAddress port:(int)port account:(NSString
*)account password:(NSString *)password
{
    TSDK_S_LOGIN_PARAM loginParam;
    memset(&loginParam, 0, sizeof(TSDK_S_LOGIN_PARAM));
    loginParam.user_id = 1;
    loginParam.auth_type = TSDK_E_AUTH_NORMAL;
    strcpy(loginParam.user_name, [account UTF8String]);
    strcpy(loginParam.password, [password UTF8String]);
    loginParam.server_type = TSDK_E_SERVER_TYPE_PORTAL;
    strcpy(loginParam.server_addr, [serverAddress UTF8String]);
    loginParam.server_port = (TSDK_UINT16)port;
    TSDK_RESULT result = tsdk_login(&loginParam);

    return result == TSDK_SUCCESS ? YES : NO;
}
```

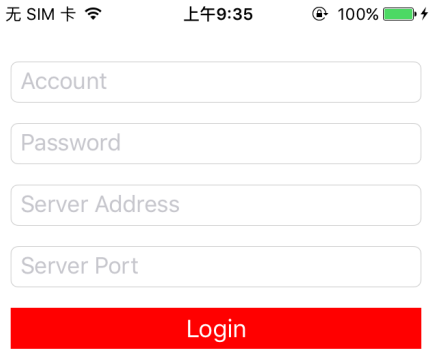
- 处理回调函数登录事件。

```
TSDK_VOID onTSDKNotifications(TSDK_UINT32 msgid, TSDK_UINT32 param1, TSDK_UINT32 param2,
TSDK_VOID *data)
{
    NSLog(@"onTUPLoginNotifications : %#x",msgid);
    dispatch_async(dispatch_get_main_queue(), ^{
        switch (msgid)
        {
            case TSDK_E_LOGIN_EVT_AUTH_SUCCESS:
            {
                NSLog(@"Uportal login success !");
                [ViewController showMessages:@"Uportal login success"];
```

```
        }
            break;
        case TSDK_E_LOGIN_EVT_AUTH_FAILED:
        {
            NSLog(@"Uportal login fail !");
            [ViewController showMessages:@"Uportal login fail"];
        }
            break;
        default:
            break;
    }
  });
}
```

4.9 编译及调试

步骤1 点击“Product > Run”，运行后如下所示。



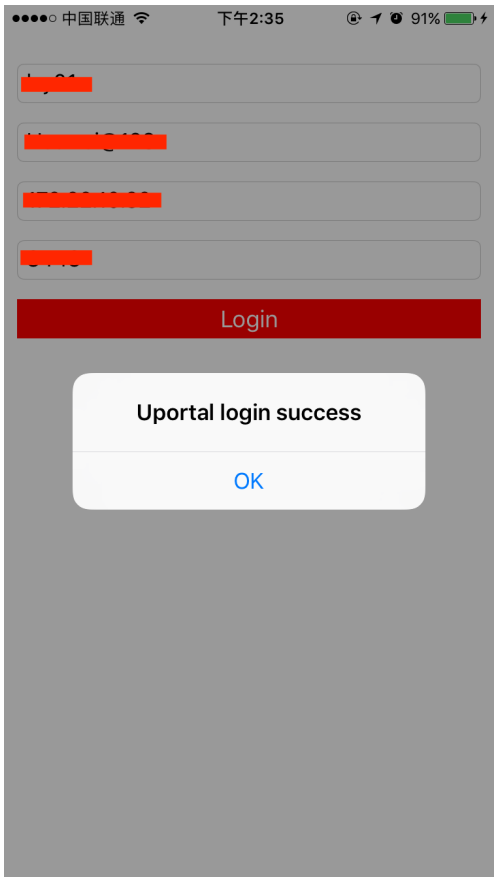
步骤2 在界面中依次输入如下表信息：

分类	输入信息	说明	备注
鉴权信息	ServerAddress	eSDK服务器地址	入参不能为空
	Server Port	eSDK服务器端口号	
	Account	eSDK登录名	
	Password	eSDK鉴权密码	入参不能为空

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。

步骤3 信息输入完成后，单击“ ”按钮。等待至页面弹出提示框，显示“Login Success！ ”，表示调用服务成功。



----结束

5 业务开发指导

5.1 业务开发总体流程

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：

步骤1 初始化业务组件；

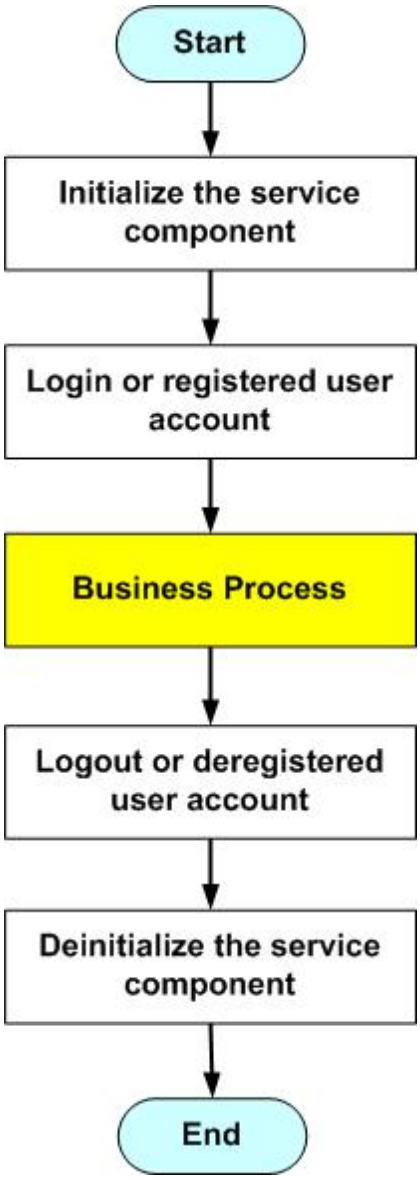
步骤2 登录用户帐号；

步骤3 业务实现调用；

步骤4 登出用户帐号；

步骤5 去初始化业务组件。

----结束



流程说明

步骤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。

步骤2 **登录用户帐号**：先完成本地地址、端口和服务器地址、端口等登录相关参数设置，再调用相应接口完成向服务器的登录。

步骤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。

步骤4 **登出用户帐号**：实现用户退出登录，确保业务接口使用的安全性。

步骤5 **去初始化组件**：实现对业务组件占用的资源释放。

----结束

5.2 组件初始化

应用场景

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

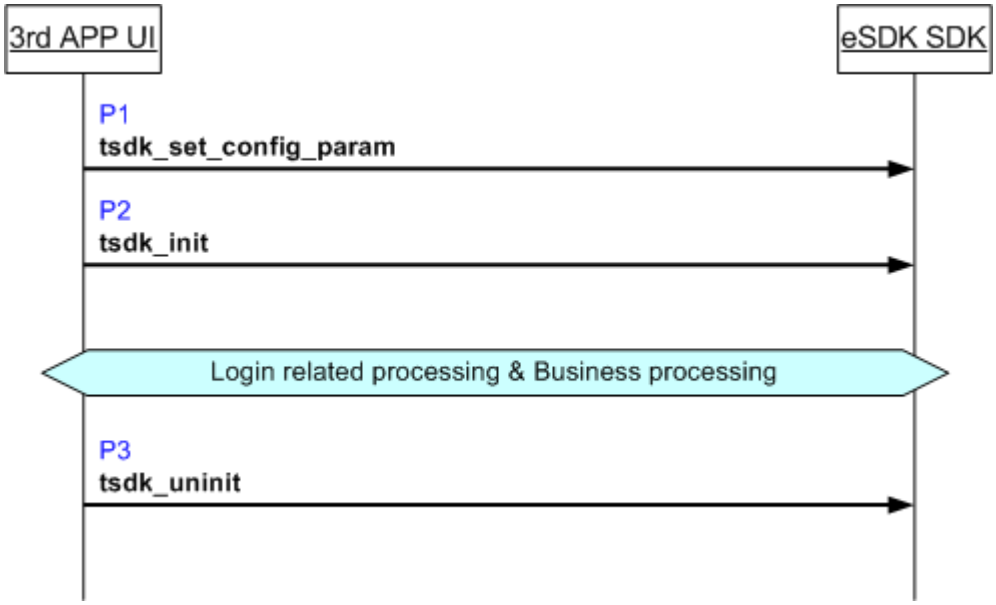
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

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tsdk_set_config_param()接口设置业务参数。

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，对应的数据结构：TSDK_S_LOG_PARAM，此参数在iOS平台必选；
2. TLS参数，对应的配置ID：TSDK_E_CONFIG_TLS_PARAM，对应的数据结构：TSDK_S_TLS_PARAM；
3. 应用程序文件路径信息，对应的配置ID：TSDK_E_CONFIG_APP_FILE_PATH_INFO，对应的数据结构：TSDK_S_APP_FILE_PATH_INFO，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，对应的数据结构：TSDK_S_DEVICE_DPI_INFO。
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：TSDK_S_CONF_CTRL_PARAM。

代码示例：

```
TSDK_S_LOG_PARAM logParam;
memset(&logParam, 0, sizeof(TSDK_S_LOG_PARAM));
NSString *path = [logPath stringByAppendingString:@"/tsdk"];
logParam.level = TSDK_E_LOG_DEBUG;
logParam.file_count = 1;
logParam.max_size_kb = 4*1024;
strcpy(logParam.path, [path UTF8String]);
TSDK_RESULT configResult = tsdk_set_config_param(TSDK_E_CONFIG_LOG_PARAM, &logParam);
```

步骤2 应用程序调用tsdk_init()接口实现组件初始化。

说明

1. 应用程序信息参数(TSDK_S_APP_INFO_PARAM)，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“SoftClient on 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. 事件通知回调函数(TSDK_FN_CALLBACK_PTR)由应用程序实现，若回调消息参数存在指针参数，则需应用程序深拷贝后使用，否则组件可能会释放资源，导致程序崩溃。
3. 应用程序关注的事件：

事件ID	事件说明
TSDK_E_LOGIN_EVT_AUTH_SUCCESS	鉴权成功(用于呈现登录过程，应用层一般无需处理)
TSDK_E_LOGIN_EVT_AUTH_FAILED	鉴权失败。
TSDK_E_LOGIN_EVT_AUTH_REFRESH_FAILED	鉴权刷新失败。
TSDK_E_LOGIN_EVT_LOGIN_SUCCESS	登录成功。
TSDK_E_LOGIN_EVT_LOGIN_FAILED	登录失败。
TSDK_E_LOGIN_EVT_LOGOUT_SUCCESS	登出成功。
TSDK_E_LOGIN_EVT_LOGOUT_FAILED	登出失败。

事件ID	事件说明
TSDK_E_LOGIN_EVT_FORCE_LOGOUT	强制登出。
TSDK_E_LOGIN_EVT_VOIP_ACCOUNT_STATUS	VOIP帐号信息。
TSDK_E_LOGIN_EVT_FIREWALL_DETECT_FAILED	防火墙探测失败。
TSDK_E_LOGIN_EVT_BUILD_STG_TUNNEL_FAILED	创建stg通道失败。
TSDK_E_CALL_EVT_CALL_START_RESULT	发起呼叫结果。
TSDK_E_CALL_EVT_CALL_INCOMING	来电事件。
TSDK_E_CALL_EVT_CALL_OUTGOING	呼出事件。
TSDK_E_CALL_EVT_CALL_RINGBACK	回铃音事件(在需要APP播放回铃音时上报)。
TSDK_E_CALL_EVT_CALL_RTP_CREATED	RTP通道已建立，可以进行二次拨号。
TSDK_E_CALL_EVT_CALL_CONNECTED	通话已建立。
TSDK_E_CALL_EVT_CALL_ENDED	呼叫结束。
TSDK_E_CALL_EVT_CALL_DESTROY	呼叫结束后销毁呼叫控制信息。
TSDK_E_CALL_EVT_OPEN_VIDEO_REQ	远端请求打开视频(音频通话升级为视频通话)。
TSDK_E_CALL_EVT_REFUSE_OPEN_VIDEO_IND	远端拒绝请求打开视频通知(远端用户拒绝或超时未响应)。
TSDK_E_CALL_EVT_CLOSE_VIDEO_IND	关闭视频通知(视频通话转为音频通话)。
TSDK_E_CALL_EVT_OPEN_VIDEO_IND	打开视频通知(音频通话转为视频通话)。
TSDK_E_CALL_EVT_REFRESH_VIEW_IND	视频view刷新通知。
TSDK_E_CALL_EVT_CALL_ROUTE_CHANGE	移动路由变化通知(主要用于iOS)。
TSDK_E_CALL_EVT_PLAY_MEDIA_END	音频文件播放结束通知。
TSDK_E_CALL_EVT_SESSION_MODIFIED	会话修改完成通知。
TSDK_E_CALL_EVT_SESSION_CODEC	会话正在使用的codec通知。
TSDK_E_CALL_EVT_HOLD_SUCCESS	保持成功。
TSDK_E_CALL_EVT_HOLD_FAILED	保持失败。

事件ID	事件说明
TSDK_E_CALL_EVT_UNHOLD_SUCCESS	恢复成功。
TSDK_E_CALL_EVT_UNHOLD_FAILED	恢复失败。
TSDK_E_CALL_EVT_ENDCALL_FAILED	结束通话失败。
TSDK_E_CALL_EVT_DIVERT_FAILED	偏转失败。
TSDK_E_CALL_EVT_BLD_TRANSFER_SUCCESS	盲转成功。
TSDK_E_CALL_EVT_BLD_TRANSFER_FAILED	盲转失败。
TSDK_E_CALL_EVT_SET_IPT_SERVICE_RESULT	登记IPT业务结果。
TSDK_E_CALL_EVT_IPT_SERVICE_INFO	ipt业务信息变更通知。
TSDK_E_CALL_EVT_AUX_DATA_RECVING	辅流接收成功
TSDK_E_CALL_EVT_AUX_DATA_STOPPED	辅流停止接收
TSDK_E_CALL_EVT_DECODE_SUCCESS	解码成功信息通知
TSDK_E_CONF_EVT_BOOK_CONF_RESULT	预约会议结果。
TSDK_E_CONF_EVT_QUERY_CONF_LIST_RESULT	查询会议列表结果。
TSDK_E_CONF_EVT_JOIN_CONF_RESULT	加入会议结果。
TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT	获取数据会议参数结果。
TSDK_E_CONF_EVT_CONFCTRL_OPERATION_RESULT	会控操作结果。
TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE	会议信息及状态状态更新。
TSDK_E_CONF_EVT_SPEAKER_IND	发言方通知。
TSDK_E_CONF_EVT_REQUEST_CONF_RIGHT_FAILED	申请会控权限失败(与会者在会议中将无会控权限，但仍可参与会议)。
TSDK_E_CONF_EVT_CONF_INCOMING_IND	会议来电通知。
TSDK_E_CONF_EVT_CONF_END_IND	会议结束通知。
TSDK_E_CONF_EVT_JOIN_DATA_CONF_RESULT	加入数据会议结果。

事件ID	事件说明
TSDK_E_CONF_EVT_AS_SCREEN_DATA_UPDATE	屏幕数据更新。
TSDK_E_CONF_EVT_AS_OWNER_CHANGE	屏幕共享权限拥有者变更通知。
TSDK_E_CONF_EVT_AS_STATE_CHANGE	屏幕共享状态变更通知。
TSDK_E_CONF_EVT_RECV_CHAT_MSG	收到会议中的聊天消息通知。
TSDK_E_CONF_EVT_PRESENTER_GIVE_IND	被设置为主讲人通知。
TSDK_E_CONF_EVT_DATA_COMPT_TOKEN_MSG	令牌权限变更通知
TSDK_E_CONF_EVT_PHONE_VIDEO_CAPABLE_IND	电话能力更新
TSDK_E_CONF_EVT_UPDATE_SHARE_MSG	共享类型更新通知
TSDK_E_CONF_EVT_AS_AUX_DEC_FRIST_FRAME	辅流解码第一帧成功的通知
TSDK_E_LDAP_FRONTSTAGE_EVT_SEARCH_RESULT	企业通讯录查询结果通知。
TSDK_E_CTD_EVT_START_CALL_RESULT	发起ctd呼叫结果。
TSDK_E_CTD_EVT_END_CALL_RESULT	结束ctd呼叫结果。
TSDK_E_CTD_EVT_CALL_STATUS_NOTIFY	ctd呼叫状态上报。
TSDK_E_EADDR_EVT_SEARCH_CONTACTS_RESULT	查询联系人结果。
TSDK_E_EADDR_EVT_SEARCH_DEPT_RESULT	查询部门结果。
TSDK_E_EADDR_EVT_GET_ICON_RESULT	获取头像结果。

代码示例：

```
TSDK_S_APP_INFO_PARAM app_info;
memset(&app_info, 0, sizeof(TSDK_S_APP_INFO_PARAM));
app_info.client_type = TSDK_E_CLIENT_MOBILE;
strcpy(app_info.product_name, "SoftClient on Mobile");
app_info.support_audio_and_video_call = TSDK_TRUE;
app_info.support_ctd = TSDK_TRUE;
app_info.support_audio_and_video_conf = TSDK_TRUE;
```



```
app_info.support_enterprise_address_book = TSDK_TRUE;
TSDK_RESULT result = tsdk_init(&app_info ,&onTSDKNotifications);
```

----结束

去初始化

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

代码示例：

```
-(BOOL)unInitLoginServer{
    TSDK_RESULT result = tsdk_uninit();
    return result == TSDK_SUCCESS ? YES : NO;
}
```

----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

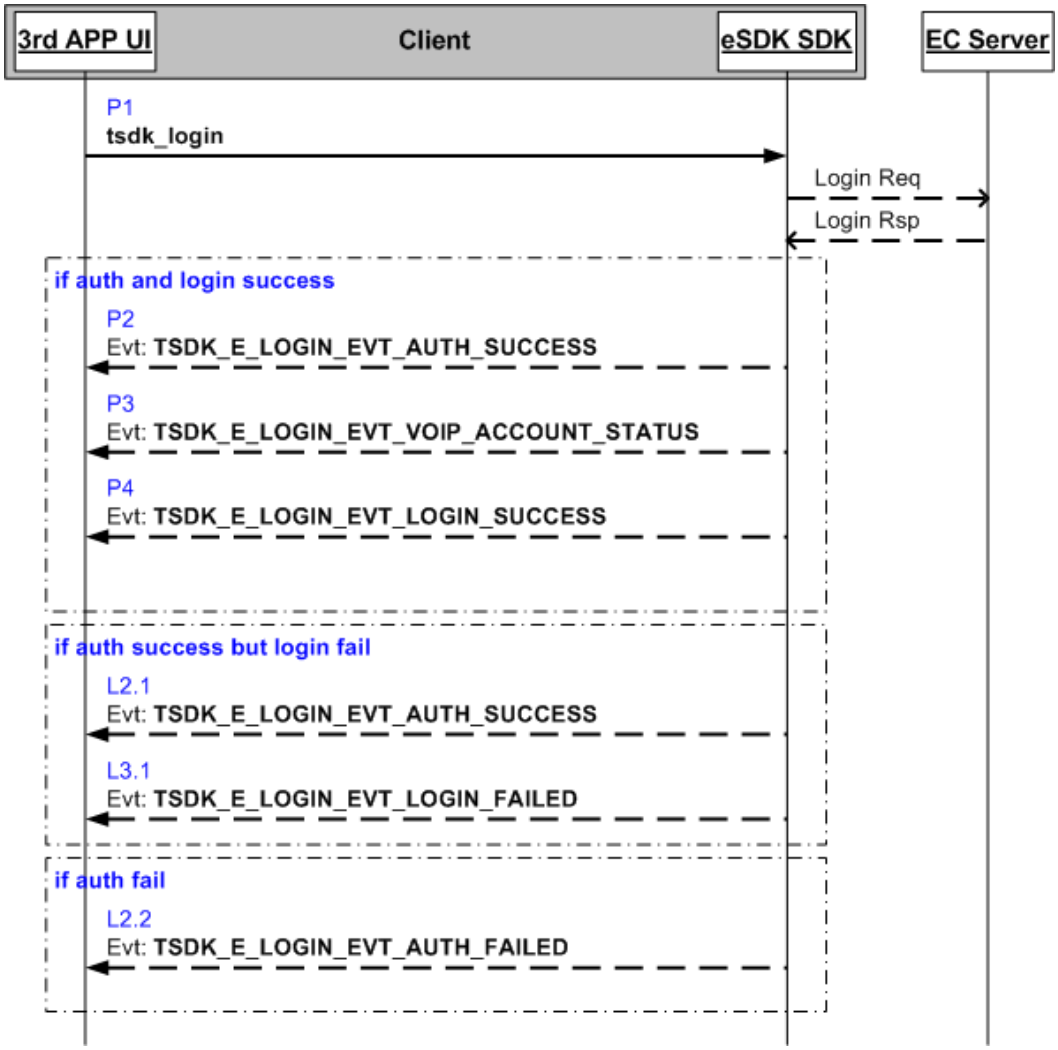
前提条件

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tsdk_login()进行登录，参数包括用户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_TYPE_PORTAL。

代码示例：

```
TSDK_S_LOGIN_PARAM loginParam;
memset(&loginParam, 0, sizeof(TSDK_S_LOGIN_PARAM));
loginParam.user_id = 1;
loginParam.auth_type = TSDK_E_AUTH_NORMAL;
strcpy(loginParam.user_name, [account UTF8String]);
strcpy(loginParam.password, [pwd UTF8String]);
loginParam.server_type = TSDK_E_SERVER_TYPE_PORTAL;
strcpy(loginParam.server_addr, [serverUrl UTF8String]);
loginParam.server_port = (TSDK_UINT16)port;
TSDK_RESULT result = tsdk_login(&loginParam);
```


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TSDK_E_LOGIN_EVT_AUTH_SUCCESS。

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。
2. 为兼容6.0 版本(本版本的前一版本)IM SDK，鉴权成功时携带IM 帐号信息参数(TSDK_S_IM_LOGIN_PARAM)，应用程序可使用此信息调用IM SDK接口完成IM登录。

代码示例：

```
case TSDK_E_LOGIN_EVT_AUTH_SUCCESS:
{
    TSDK_S_IM_LOGIN_PARAM *im_login_parama = (TSDK_S_IM_LOGIN_PARAM
*)notify.data;
    LoginServerInfo *LoginAccessServer = [[LoginServerInfo alloc] init];
    LoginAccessServer.eserverUri = [NSString stringWithUTF8String:im_login_parama->e_server_uri];
    LoginAccessServer.maaUri = [NSString stringWithUTF8String:im_login_parama->maa_server_uri];
    LoginAccessServer.sipAccount = [NSString stringWithUTF8String:im_login_parama->account];
    LoginAccessServer.sipPwd= [NSString stringWithUTF8String:im_login_parama->password];
    LoginAccessServer.token = [NSString stringWithUTF8String:im_login_parama->token];
    self.loginServerInfo = LoginAccessServer;
    DDLogInfo(@"authorize success");
    break;
}
```

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TSDK_E_LOGIN_EVT_VOIP_ACCOUNT_STATUS。

说明

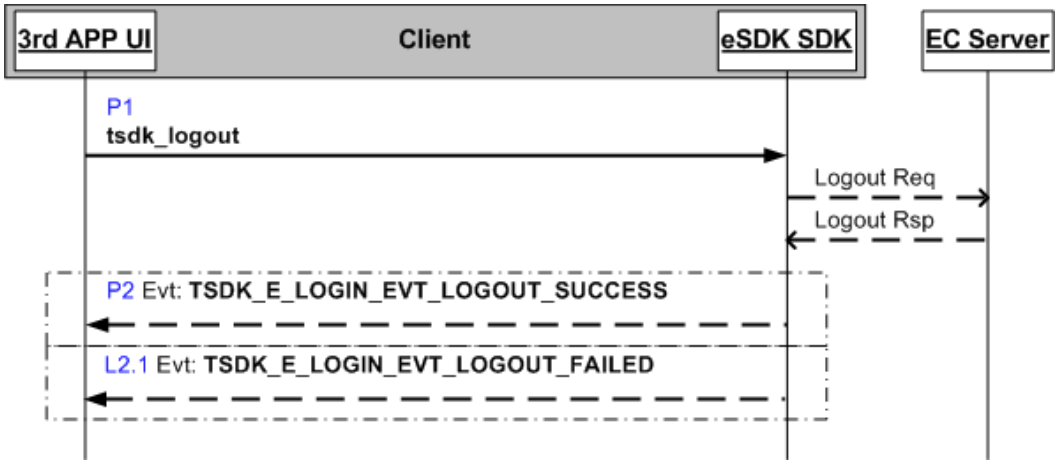
1. 如果登录成功，会上报账号短号号码，UI应保存此号码，以方便后续操作。

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

----结束

用户主动注销

图 5-3 注销处理流程



步骤1 UI调用tsdk_logout()发起注销。

代码示例：

```
-(BOOL)logout
{
    TSDK_RESULT ret = tsdk_logout();
    BOOL result = (TSDK_SUCCESS == ret) ? YES : NO;
    return result;
}
```

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

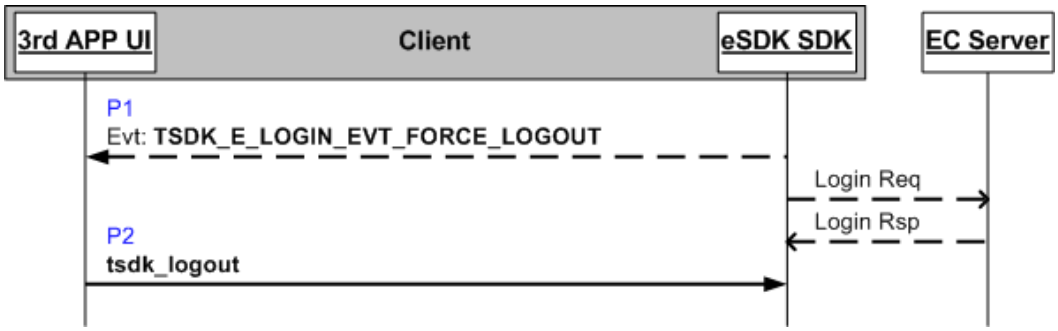
代码示例：

```
case TSDK_E_LOGIN_EVT_LOGOUT_SUCCESS:
{
    sipStatus = kCallSipStatusUnRegistered;
    [self isSipRegistered:sipStatus];
    break;
}
```

----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TSDK_E_LOGIN_EVT_FORCE_LOGOUT。

步骤2 UI调用tsdk_logout()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。

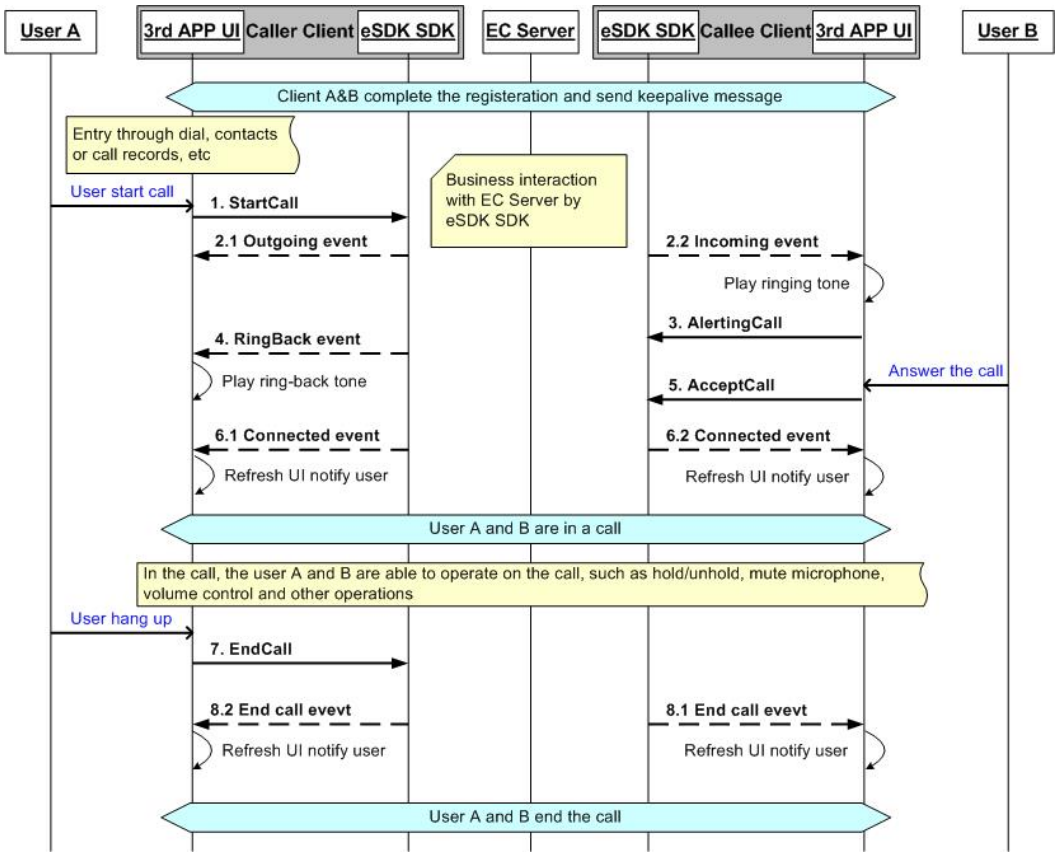
5.4 音视频呼叫

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VC 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VC 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。

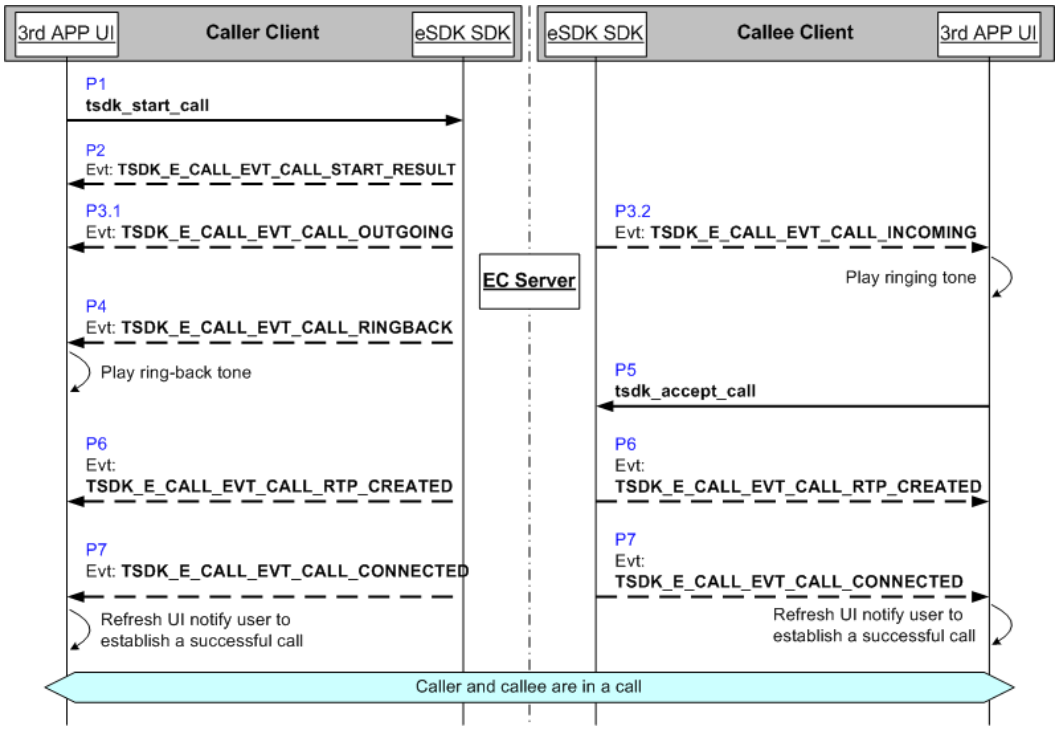
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

主被叫客户端均已注册。

流程说明

图 5-5 建立音频通话流程



步骤1 主叫UI调用tsdk_start_call()接口发起音频呼叫，传入参数包含被叫号码和是否发起视频呼叫，返回参数call_id标识本次呼叫ID。

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：

```
TSDK_BOOL isVideo = ((TSDK_CALL_E_CALL_TYPE)callType==CALL_VIDEO)?TSDK_TRUE:TSDK_FALSE;
TSDK_UINT32 callid = 0;
TSDK_RESULT ret = tsdk_start_call(&callid,(TSDK_CHAR*)[number UTF8String], isVideo);
```

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TSDK_E_CALL_EVT_CALL_START_RESULT。

步骤3 主叫SDK发出呼叫请求消息，向UI上报呼出事件TSDK_E_CALL_EVT_CALL_OUTGOING；被叫SDK收到呼叫请求消息，向UI上报来电事件TSDK_E_CALL_EVT_CALL_INCOMING。通知事件携带参数callid标识本次呼叫，对应的事件数据结构中TSDK_S_CALL_INFO包含远端号码、呼叫类型和呼叫状态等呼叫信息。

代码示例：

```
case TSDK_E_CALL_EVT_CALL_INCOMING:
{
    TSDK_S_CALL_INFO *callInfo = (TSDK_S_CALL_INFO *)notify.data;
    CallInfo *tsdkCallInfo = [CallInfo transfromFromCallInfoStract:callInfo];
    [self resetUCVideoOrientAndIndexWithCallId:0];
    NSString *callId = [NSString stringWithFormat:@"%d", callInfo->call_id];
    [_tsdkCallInfoDic setObject:tsdkCallInfo forKey:callId];
    NSDictionary *resultInfo = @{
        TSDK_CALL_INFO_KEY : tsdkCallInfo
```



```
    };
    [self respondsCallDelegateWithType:CALL_INCOMMING result:resultInfo];
}
```

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

代码示例：

```
case TSDK_E_CALL_EVT_CALL_RINGBACK:
{
    NSDictionary *resultInfo = @{
        TSDK_CALL_RINGBACK_KEY : [NSNumber numberWithBool:true]
    };
    [self respondsCallDelegateWithType:CALL_RINGBACK result:resultInfo];
    break;
}
```

步骤5 被叫调用tsdk_accept_call()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：

```
- (BOOL) answerComingCallType:(TUP_CALL_TYPE)callType callId:(unsigned int)callId
{
    TSDK_RESULT ret = tsdk_accept_call((TSDK_UINT32)callId, callType == CALL_AUDIO ? TSDK_FALSE :
TSDK_TRUE);
    DDLogInfo(@"Call_Log:answer call type is %d,result is %d, callid: %d",callType,ret,callId);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

步骤6 主被叫SDK向UI上报通话建立事件TSDK_E_CALL_EVT_CALL_RTP_CREATED。

说明

RTP通道已建立，可以进行一些二次拨号等操作。

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

代码示例：

```
case TSDK_E_CALL_EVT_CALL_CONNECTED:
{
    DDLogInfo(@"Call_Log: recv call notify :CALL_E_EVT_CALL_CONNECTED");
    TSDK_S_CALL_INFO *callInfo = (TSDK_S_CALL_INFO *)notify.data;
    CallInfo *tsdkCallInfo = [CallInfo transfromFromCallInfoStract:callInfo];
    NSString *callId = [NSString stringWithFormat:@"%d", tsdkCallInfo.stateInfo.callId];
    [_tsdkCallInfoDic setObject:tsdkCallInfo forKey:callId];
    NSDictionary *resultInfo = @{
        TSDK_CALL_INFO_KEY : tsdkCallInfo
    };
    [self respondsCallDelegateWithType:CALL_CONNECT result:resultInfo];
}
```

----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。

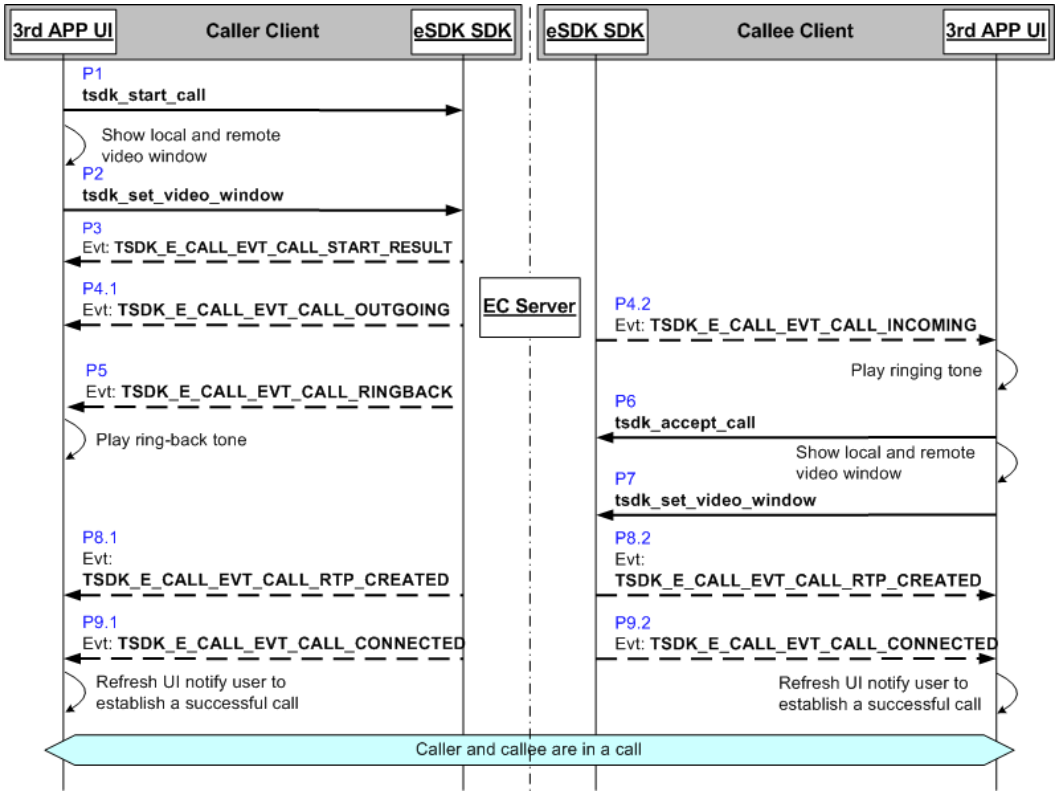
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

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tsdk_start_call()接口发起视频呼叫，传入参数包含被叫号码和是否发起视频呼叫，返回参数call_id标识本次呼叫ID。

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：

```
TSDK_BOOL isVideo = ((TSDK_CALL_E_CALL_TYPE)callType==CALL_VIDEO)?TSDK_TRUE:TSDK_FALSE;
TSDK_UINT32 callid = 0;
TSDK_RESULT ret = tsdk_start_call(&callid,(TSDK_CHAR*)[number UTF8String], isVideo);
```

步骤2 主叫发起视频呼叫后，就调用tsdk_set_video_window()接口设置视频窗口与呼叫的绑定关系。

代码示例：

```
- (BOOL)updateVideoWindowWithLocal:(id)localVideoView andRemote:(id)remoteVideoView andBFCP:
(id)bfcpVideoView callId:(unsigned int)callId
{
    TSDK_S_VIDEO_WND_INFO videoInfo[3];
    memset_s(videoInfo, sizeof(TSDK_S_VIDEO_WND_INFO) * 2, 0, sizeof(TSDK_S_VIDEO_WND_INFO) * 2);
    videoInfo[0].video_wnd_type = TSDK_E_VIDEO_WND_LOCAL;
    videoInfo[0].render = (TSDK_UPTR)localVideoView;
    videoInfo[0].display_mode = TSDK_E_VIDEO_WND_DISPLAY_FULL;
    videoInfo[1].video_wnd_type = TSDK_E_VIDEO_WND_REMOTE;
    videoInfo[1].render = (TSDK_UPTR)remoteVideoView;
    videoInfo[1].display_mode = TSDK_E_VIDEO_WND_DISPLAY_CUT;
    videoInfo[2].video_wnd_type = TSDK_E_VIDEO_WND_AUX_DATA;
    videoInfo[2].render = (TSDK_UPTR)bfcpVideoView;
    TSDK_RESULT ret;
    videoInfo[2].display_mode = TSDK_E_VIDEO_WND_DISPLAY_CUT;
    ret = tsdk_set_video_window((TSDK_UINT32)callId, 3, videoInfo);
    DDLogInfo(@"Call_Log: tsdk_set_video_window = %d",ret);
    [self updateVideoRenderInfoWithVideoIndex:CameraIndexFront
withRenderType:TSDK_E_VIDEO_WND_LOCAL andCallId:callId];
    [self updateVideoRenderInfoWithVideoIndex:CameraIndexFront
withRenderType:TSDK_E_VIDEO_WND_REMOTE andCallId:callId];    return (TSDK_SUCCESS == ret);
}
```

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TSDK_E_CALL_EVT_CALL_START_RESULT。

步骤4 主叫SDK发出呼叫请求消息，向UI上报呼出事件TSDK_E_CALL_EVT_CALL_OUTGOING；被叫SDK收到呼叫请求消息，向UI上报来电事件TSDK_E_CALL_EVT_CALL_INCOMING，通知事件携带参数call_id标识本次呼叫，对应的事件数据结构中TSDK_S_CALL_INFO包含对端号码、是否是视频呼叫和呼叫状态等呼叫信息。

代码示例：

```
case TSDK_E_CALL_EVT_CALL_INCOMING:
{
    TSDK_S_CALL_INFO *callInfo = (TSDK_S_CALL_INFO *)notify.data;
    CallInfo *tsdkCallInfo = [CallInfo transfromFromCallInfoStract:callInfo];
    [self resetUCVideoOrientAndIndexWithCallId:0];
    NSString *callId = [NSString stringWithFormat:@"%d", callInfo->call_id];
    [_tsdkCallInfoDic setObject:tsdkCallInfo forKey:callId];
    NSDictionary *resultInfo = @{
        TSDK_CALL_INFO_KEY : tsdkCallInfo
    };
    [self respondsCallDelegateWithType:CALL_INCOMMING result:resultInfo];
}
```

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

代码示例：

```
case TSDK_E_CALL_EVT_CALL_RINGBACK:
{
    NSDictionary *resultInfo = @{
        TSDK_CALL_RINGBACK_KEY : [NSNumber numberWithBool:true]
    };
    [self respondsCallDelegateWithType:CALL_RINGBACK result:resultInfo];
    break;
}
```


步骤6 被叫调用tsdk_accept_call()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：

```
- (BOOL) answerComingCallType:(TUP_CALL_TYPE)callType callId:(unsigned int)callId
{
    TSDK_RESULT ret = tsdk_accept_call((TSDK_UINT32)callId, callType == CALL_AUDIO ? TSDK_FALSE : TSDK_TRUE);
    DDLogInfo(@"Call_Log:answer call type is %d,result is %d, callid: %d",callType,ret,callId);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

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tsdk_set_video_window()接口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上报通话建立事件TSDK_E_CALL_EVT_CALL_RTP_CREATED。

说明

RTP通道已建立，可以进行一些二次拨号等操作。

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

代码示例：

```
case TSDK_E_CALL_EVT_CALL_CONNECTED:
{
    DDLogInfo(@"Call_Log: recv call notify :CALL_E_EVT_CALL_CONNECTED");
    TSDK_S_CALL_INFO *callInfo = (TSDK_S_CALL_INFO *)notify.data;
    CallInfo *tsdkCallInfo = [CallInfo transfromFromCallInfoStract:callInfo];
    NSString *callId = [NSString stringWithFormat:@"%d", tsdkCallInfo.stateInfo.callId];
    [_tsdkCallInfoDic setObject:tsdkCallInfo forKey:callId];
    NSDictionary *resultInfo = @{
        TSDK_CALL_INFO_KEY : tsdkCallInfo
    };
    [self respondsCallDelegateWithType:CALL_CONNECT result:resultInfo];
}
```

----结束

注意事项

无。

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	

eSDK CloudVC 20.1.RC1 开发指南(iOS)

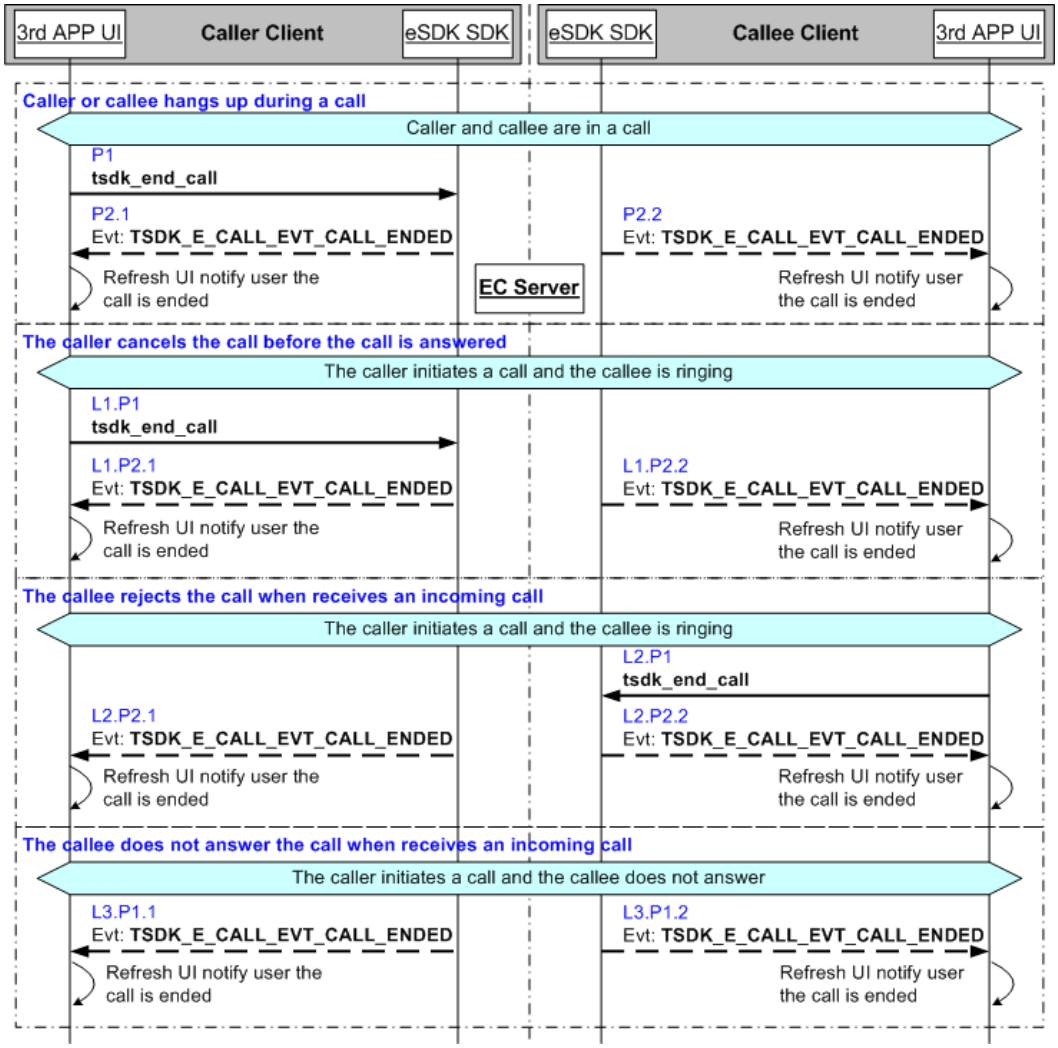
变更记录	eSDK版本	解决方案版本	变更内容
变更	NA	NA	NA

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tsdk_end_call()接口挂断通话。

代码示例：

```
-(BOOL)endCall:(unsigned int)callId
{
    TSDK_UINT32 callid = (TSDK_UINT32)callId;
    TSDK_RESULT ret = tsdk_end_call(callid);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

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TSDK_E_CALL_EVT_CALL_ENDED，UI刷新界面显示通话结束。

代码示例：

```
case TSDK_E_CALL_EVT_CALL_ENDED:
{
    DDLogInfo(@"Call_Log: recv call notify :CALL_E_EVT_CALL_ENDED");
    TSDK_S_CALL_INFO *callInfo = (TSDK_S_CALL_INFO *)notify.data;
    CallInfo *tsdkCallInfo = [CallInfo transfromFromCallInfoStract:callInfo];
    NSDictionary *resultInfo = @{
        TSDK_CALL_INFO_KEY : tsdkCallInfo
    };
    [self respondsCallDelegateWithType:CALL_CLOSE result:resultInfo];
}
```

----结束

主叫在通话接通前取消呼叫

步骤1 主叫UI调用tsdk_end_call()接口取消呼叫。

代码示例：

```
-(BOOL)endCall:(unsigned int)callId
{
    TSDK_UINT32 callid = (TSDK_UINT32)callId;
    TSDK_RESULT ret = tsdk_end_call(callid);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

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TSDK_E_CALL_EVT_CALL_ENDED，UI刷新界面显示通话结束。

代码示例：

```
case TSDK_E_CALL_EVT_CALL_ENDED:
{
    DDLogInfo(@"Call_Log: recv call notify :CALL_E_EVT_CALL_ENDED");
    TSDK_S_CALL_INFO *callInfo = (TSDK_S_CALL_INFO *)notify.data;
    CallInfo *tsdkCallInfo = [CallInfo transfromFromCallInfoStract:callInfo];
    NSDictionary *resultInfo = @{
        TSDK_CALL_INFO_KEY : tsdkCallInfo
    };
    [self respondsCallDelegateWithType:CALL_CLOSE result:resultInfo];
}
```

----结束

被叫在收到来电时拒绝呼叫

步骤1 被叫UI调用tsdk_end_call()接口拒绝通话。

代码示例：

```
-(BOOL)endCall:(unsigned int)callId
{
    TSDK_UINT32 callid = (TSDK_UINT32)callId;
    TSDK_RESULT ret = tsdk_end_call(callid);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```


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TSDK_E_CALL_EVT_CALL_ENDED，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TSDK_E_CALL_EVT_CALL_CONNECTED事件，需要用户通过主叫UI主动挂断呼叫。

代码示例：

```
case TSDK_E_CALL_EVT_CALL_ENDED:
{
    DDLogInfo(@"Call_Log: recv call notify :CALL_E_EVT_CALL_ENDED");
    TSDK_S_CALL_INFO *callInfo = (TSDK_S_CALL_INFO *)notify.data;
    CallInfo *tsdkCallInfo = [CallInfo transfromFromCallInfoStract:callInfo];
    NSDictionary *resultInfo = @{
        TSDK_CALL_INFO_KEY : tsdkCallInfo
    };
    [self respondsCallDelegateWithType:CALL_CLOSE result:resultInfo];
}
```

----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TSDK_E_CALL_EVT_CALL_ENDED，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TSDK_E_CALL_EVT_CALL_RTP_CREATED或TSDK_E_CALL_EVT_CALL_CONNECTED事件，需要用户通过主叫UI主动挂断呼叫。

代码示例：

```
case TSDK_E_CALL_EVT_CALL_ENDED:
{
    DDLogInfo(@"Call_Log: recv call notify :CALL_E_EVT_CALL_ENDED");
    TSDK_S_CALL_INFO *callInfo = (TSDK_S_CALL_INFO *)notify.data;
    CallInfo *tsdkCallInfo = [CallInfo transfromFromCallInfoStract:callInfo];
    NSDictionary *resultInfo = @{
        TSDK_CALL_INFO_KEY : tsdkCallInfo
    };
    [self respondsCallDelegateWithType:CALL_CLOSE result:resultInfo];
}
```

----结束

注意事项

无。

5.4.5 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。

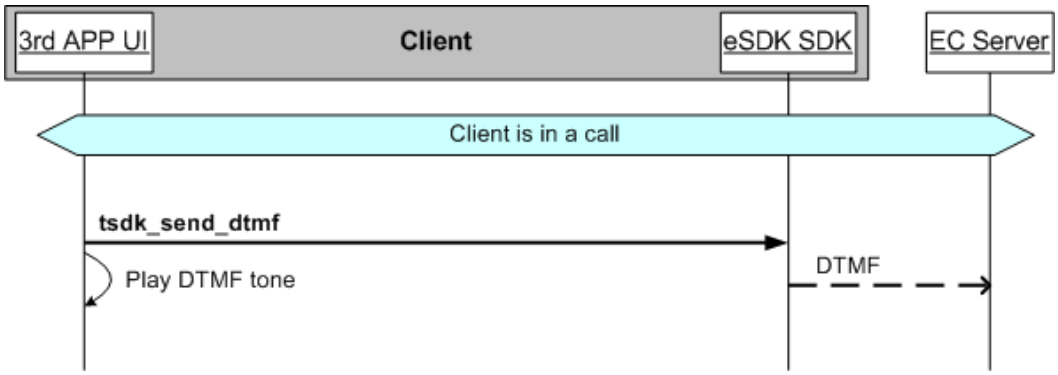
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

已建立与业务服务器间的通话。

流程说明

图 5-8 二次拨号流程



步骤1 UI调用接口tsdk_send_dtmf()在通话中发送DTMF信号，参数为呼叫标识callid和需要发送的DTMF信号值TSDK_E_DTMF_TONE；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：

```
- (BOOL)sendDTMFWithDialNum:(NSString *)number callId:(unsigned int)callId
{
    TSDK_E_DTMF_TONE dtmfTone = (TSDK_E_DTMF_TONE)[number intValue];
    if ([number isEqualToString:@"*"])
    {
        dtmfTone = TSDK_E_DTMF_STAR;
    }
    else if ([number isEqualToString:@"#"])
    {
        dtmfTone = TSDK_E_DTMF_POUND;
    }
    TSDK_UINT32 callid = callId;
    TSDK_RESULT ret = tsdk_send_dtmf((TSDK_UINT32)callid,(TSDK_E_DTMF_TONE)dtmfTone);
    DDLogInfo(@"Call_Log: tsdk_send_dtmf = %@",(TSDK_SUCCESS == ret)?@"YES":@"NO");
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

----结束

注意事项

无。

5.4.6 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。

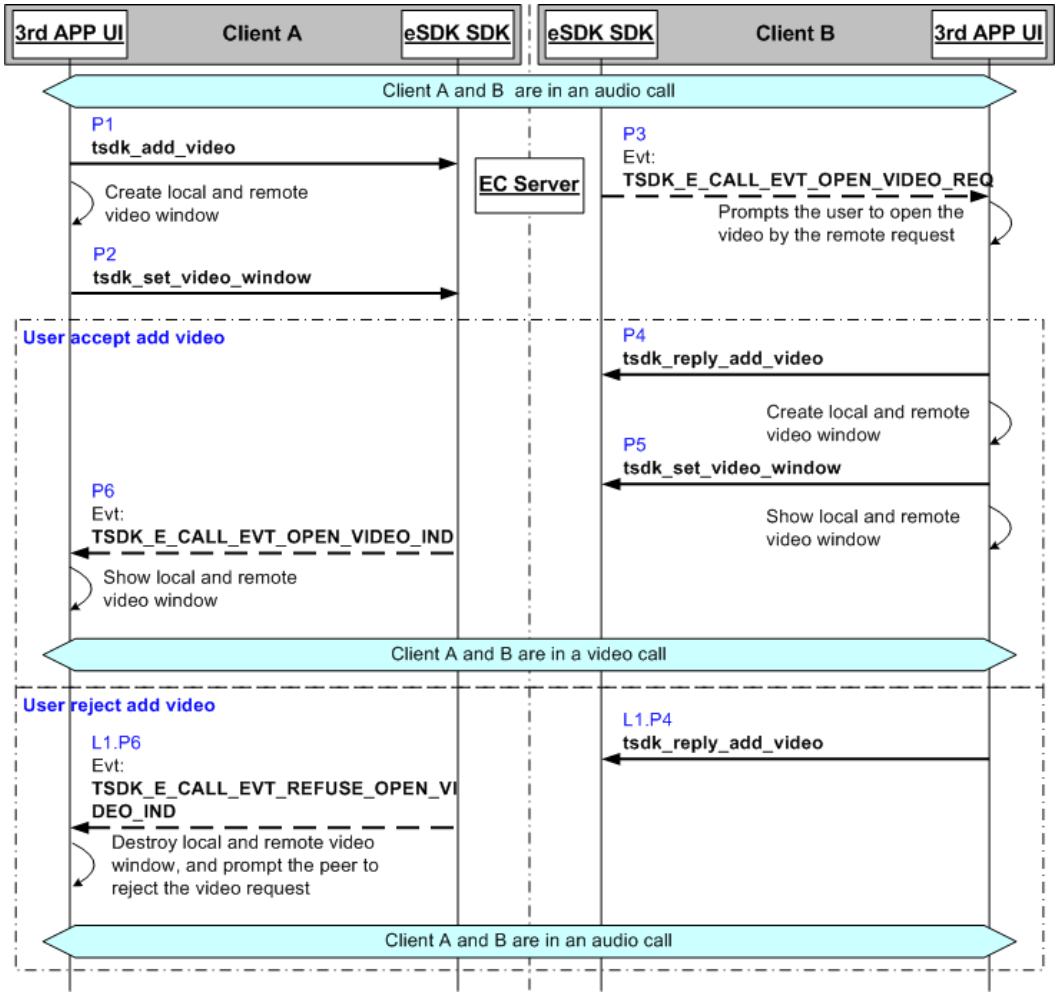
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

音频通话已建立，主被叫正在通话中。

流程说明

图 5-9 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调用tsdk_add_video()发起音频转视频呼叫请求。

代码示例：

```
-(BOOL)upgradeAudioToVideoCallWithCallId:(unsigned int)callId
{
    TSDK_RESULT ret = tsdk_add_video((TSDK_UINT32)callId);
    DDLogInfo(@"Call_Log: tsdk_add_video = %d",ret);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

步骤2 请求发起方UI先完成本地窗口和远端窗口创建，再调用tsdk_set_video_window()接口设置视频窗口与呼叫的绑定关系。

代码示例：

```
- (BOOL)updateVideoWindowWithLocal:(id)localVideoView andRemote:(id)remoteVideoView andBFCP:(id)bfcpVideoView callId:(unsigned int)callId
{
    TSDK_S_VIDEO_WND_INFO videoInfo[3];
```

```
    memset_s(videoInfo, sizeof(TSDK_S_VIDEO_WND_INFO) * 2, 0, sizeof(TSDK_S_VIDEO_WND_INFO) * 2);
    videoInfo[0].video_wnd_type = TSDK_E_VIDEO_WND_LOCAL;
    videoInfo[0].render = (TSDK_UPTR)localVideoView;
    videoInfo[0].display_mode = TSDK_E_VIDEO_WND_DISPLAY_FULL;
    videoInfo[1].video_wnd_type = TSDK_E_VIDEO_WND_REMOTE;
    videoInfo[1].render = (TSDK_UPTR)remoteVideoView;
    videoInfo[1].display_mode = TSDK_E_VIDEO_WND_DISPLAY_CUT;
    videoInfo[2].video_wnd_type = TSDK_E_VIDEO_WND_AUX_DATA;
    videoInfo[2].render = (TSDK_UPTR)bfcpVideoView;
    TSDK_RESULT ret;
    videoInfo[2].display_mode = TSDK_E_VIDEO_WND_DISPLAY_CUT;
    ret = tsdk_set_video_window((TSDK_UINT32)callId, 3, videoInfo);
    DDLogInfo(@"Call_Log: tsdk_set_video_window = %d",ret);
    [self updateVideoRenderInfoWithVideoIndex:CameraIndexFront
withRenderType:TSDK_E_VIDEO_WND_LOCAL andCallId:callId];
    [self updateVideoRenderInfoWithVideoIndex:CameraIndexFront
withRenderType:TSDK_E_VIDEO_WND_REMOTE andCallId:callId];    return (TSDK_SUCCESS == ret);
}
```

步骤3 被请求方SDK收到请求后，向UI上报对方请求增加视频事件TSDK_E_CALL_EVT_OPEN_VIDEO_REQ，UI应刷新界面通知用户远端请求转视频。

代码示例：

```
case TSDK_E_CALL_EVT_OPEN_VIDEO_REQ:
{
    NSString *callId = [NSString stringWithFormat:@"%d",notify.param1];
    NSDictionary *callUpgradePassiveInfo = [NSDictionary
dictionaryWithObjectsAndKeys:callId,CALL_ID,nil];
    [self respondsCallDelegateWithType:CALL_UPGRADE_VIDEO_PASSIVE
result:callUpgradePassiveInfo];
    DDLogInfo(@"Call_Log: call revice CALL_E_EVT_CALL_ADD_VIDEO");
    break;
}
```

步骤4 被请求方UI调用tsdk_reply_add_video()接口来处理转视频请求，参数为呼叫标识call_id和is_accept标识。接收转视频请求时，is_accept标识的值为TSDK_TRUE；拒绝转视频请求时，is_accept标识的值为TSDK_FALSE。

代码示例：

```
-(BOOL)replyAddVideoCallIsAccept:(BOOL)accept callId:(unsigned int)callId
{
    TSDK_BOOL isAccept = accept;
    TSDK_RESULT ret = tsdk_reply_add_video((TSDK_UINT32)callId , isAccept);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

步骤5 被请求用户接受转视频请求，UI先完成本地窗口和远端窗口创建，再调用tsdk_set_video_window()接口设置视频窗口与呼叫的绑定关系方。

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。

代码示例：

```
- (BOOL)updateVideoWindowWithLocal:(id)localVideoView andRemote:(id)remoteVideoView andBFCP:
(id)bfcpVideoView callId:(unsigned int)callId
{
    TSDK_S_VIDEO_WND_INFO videoInfo[3];
    memset_s(videoInfo, sizeof(TSDK_S_VIDEO_WND_INFO) * 2, 0, sizeof(TSDK_S_VIDEO_WND_INFO) * 2);
    videoInfo[0].video_wnd_type = TSDK_E_VIDEO_WND_LOCAL;
    videoInfo[0].render = (TSDK_UPTR)localVideoView;
    videoInfo[0].display_mode = TSDK_E_VIDEO_WND_DISPLAY_FULL;
    videoInfo[1].video_wnd_type = TSDK_E_VIDEO_WND_REMOTE;
    videoInfo[1].render = (TSDK_UPTR)remoteVideoView;
    videoInfo[1].display_mode = TSDK_E_VIDEO_WND_DISPLAY_CUT;
    videoInfo[2].video_wnd_type = TSDK_E_VIDEO_WND_AUX_DATA;
```

```
    videoInfo[2].render = (TSDK_UPTR)bfcpVideoView;
    TSDK_RESULT ret;
    videoInfo[2].display_mode = TSDK_E_VIDEO_WND_DISPLAY_CUT;
    ret = tsdk_set_video_window((TSDK_UINT32)callId, 3, videoInfo);
    DDLogInfo(@"Call_Log: tsdk_set_video_window = %d",ret);
    [self updateVideoRenderInfoWithVideoIndex:CameraIndexFront
withRenderType:TSDK_E_VIDEO_WND_LOCAL andCallId:callId];
    [self updateVideoRenderInfoWithVideoIndex:CameraIndexFront
withRenderType:TSDK_E_VIDEO_WND_REMOTE andCallId:callId];    return (TSDK_SUCCESS == ret);
}
```

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，请求方向UI上报打开视频通知事件TSDK_E_CALL_EVT_OPEN_VIDEO_IND，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，UI显示远端和近端视频窗口。

若被请求方拒绝视频请求，请求方向UI上报远端拒绝请求打开视频通知事件TSDK_E_CALL_EVT_REFUSE_OPEN_VIDEO_IND，UI销毁远端和近端视频窗口。

代码示例：

```
case TSDK_E_CALL_EVT_OPEN_VIDEO_IND:
{
    NSString *callId = [NSString stringWithFormat:@"%d",notify.param1];
    NSDictionary *callUpgradePassiveInfo = [NSDictionary
dictionaryWithObjectsAndKeys:callId,CALL_ID,nil];
    [self respondsCallDelegateWithType:CALL_REFUSE_OPEN_VIDEO result:callUpgradePassiveInfo];
    DDLogInfo(@"Call_Log: call CALL_E_EVT_CALL_DEL_VIDEO");
    break;
}
```

----结束

注意事项

无。

5.4.7 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。

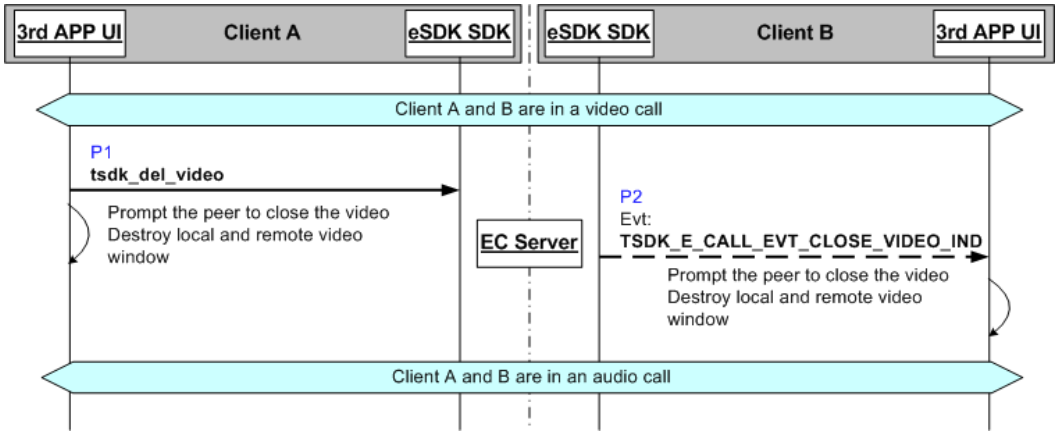
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

视频通话已建立，主被叫正在通话中。

流程说明

图 5-10 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tsdk_del_video()接口发起视频转音频呼叫请求。请求方关闭摄像头，销毁远端和近端视频窗口，刷新界面。

代码示例：

```
-(BOOL)downgradeVideoToAudioCallWithCallId:(unsigned int)callId
{
    TSDK_RESULT ret = tsdk_del_video((TSDK_UINT32)callId);
    DDLogInfo(@"Call_Log: tsdk_del_video = %d",ret);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

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，并向UI上报关闭视频通知事件TSDK_E_CALL_EVT_CLOSE_VIDEO_IND，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：

```
case TSDK_E_CALL_EVT_CLOSE_VIDEO_IND:
{
    NSString *callId = [NSString stringWithFormat:@"%d",notify.param1];
    NSDictionary *callDowngradePassiveInfo = [NSDictionary
dictionaryWithObjectsAndKeys:callId,CALL_ID,nil];
    [self respondsCallDelegateWithType:CALL_DOWNGRADE_VIDEO_PASSIVE
result:callDowngradePassiveInfo];
    DDLogInfo(@"Call_Log: call CALL_E_EVT_CALL_DEL_VIDEO");
    break;
}
```

----结束

注意事项

无。

5.4.8 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。

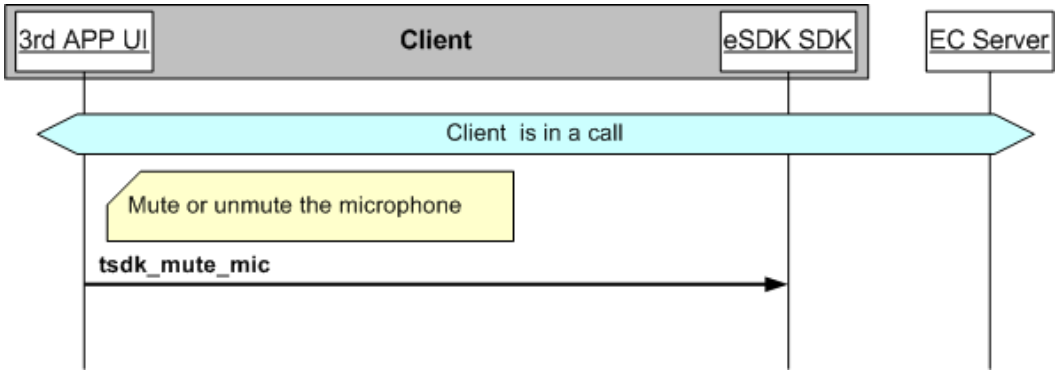
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

通话已建立，主被叫正在通话中。

流程说明

图 5-11 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tsdk_mute_mic()接口关闭或打开麦克风，参数call_id标识指定呼叫，关闭麦克风时is_mute为TSDK_TRUE，打开麦克风时is_mute为TSDK_FALSE。

代码示例：

```
-(BOOL)muteMic:(BOOL)mute callId:(unsigned int)callId
{
    TSDK_RESULT result = tsdk_mute_mic(callId , mute);
    return result == TSDK_SUCCESS ? YES : NO;
}
```

----结束

注意事项

无。

5.4.9 通话中暂停视频

应用场景

用户在视频通话中暂停或恢复视频发送。

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。

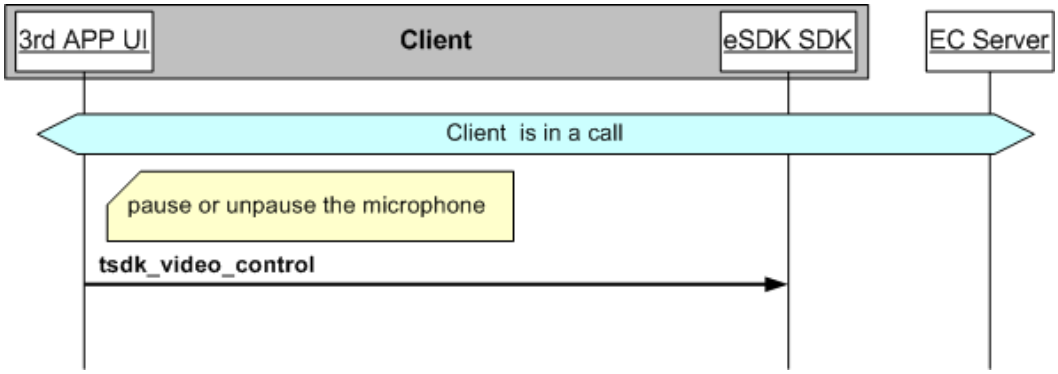
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

视频通话已建立，主被叫正在通话中。

流程说明

图 5-12 通话中暂停视频采集流程



说明

通过操作本地媒体来设置和取消暂停视频采集。

步骤1 UI调用tsdk_video_control()接口暂停或继续视频。

代码示例：

```
-(void)videoControlWithCmd:(EN_VIDEO_OPERATION)control andModule:
(EN_VIDEO_OPERATION_MODULE)module andIsSync:(BOOL)isSync callId:(unsigned int)callId
{
    TSDK_S_VIDEO_CTRL_INFO videoControlInfos;
    memset_s(&videoControlInfos, sizeof(TSDK_S_VIDEO_CTRL_INFO), 0,
sizeof(TSDK_S_VIDEO_CTRL_INFO));
```

```
    TSDK_UINT32 call_id = (TSDK_UINT32)callId;
    videoControlInfos.object = module;
    videoControlInfos.operation = control;
    videoControlInfos.is_sync = isSync;
    TSDK_RESULT ret = tsdk_video_control(call_id, &videoControlInfos);
}
```

----结束

注意事项

无。

5.4.10 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tsdk_get_devices()获取音频视频设备列表。

说明

- 获取麦克风列表时，device_type值为TSDK_E_DEVICE_MIC。
- 获取扬声器列表时，device_type值为TSDK_E_DEVICE_SPEAKER。
- 获取摄像头列表时，device_type值为TSDK_E_DEVICE_CAMERA。

代码示例：

```
-(BOOL)obtainDeviceListWityType:(DEVICE_TYPE)deviceType
{
    TSDK_UINT32 deviceNum = 0;
    TSDK_S_DEVICE_INFO *deviceInfo = nullptr;
    memset(deviceInfo, 0, sizeof(TSDK_S_DEVICE_INFO));
    TSDK_RESULT ret = tsdk_get_devices((TSDK_E_DEVICE_TYPE)deviceType, &deviceNum, deviceInfo);
    DDLogInfo(@"Call_Log: tsdk_get_devices = %#x,count:%d",ret,deviceNum);
```

```
    if (deviceNum>0)
    {
        DDLogInfo(@"again");
        deviceInfo = new TSDK_S_DEVICE_INFO[deviceNum];
        TSDK_RESULT rets = tsdk_get_devices((TSDK_E_DEVICE_TYPE)deviceType, &deviceNum,
deviceInfo);
        DDLogInfo(@"Call_Log: tsdk_get_devices = %#x,count:%d",rets,deviceNum);
        for (int i = 0; i<deviceNum; i++)
       {
            DDLogInfo(@"Call_Log: ulIndex:%d,strName:%s,string:
%@",deviceInfo[i].index,deviceInfo[i].device_name,[NSString
stringWithUTF8String:deviceInfo[i].device_name]);        }
    }
    delete [] deviceInfo;
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

----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tsdk_set_mobile_audio_route()设置移动音频路由设备。

说明

移动的音频设备包括：蓝牙、听筒和耳机

代码示例：

```
-(BOOL)configAudioRoute:(ROUTE_TYPE)route
{
    TSDK_E_MOBILE_AUIDO_ROUTE audioRoute = (TSDK_E_MOBILE_AUIDO_ROUTE)route;
    TSDK_RESULT result = tsdk_set_mobile_audio_route(audioRoute);
    DDLogInfo(@"tsdk_set_mobile_audio_route result is %@, audioRoute is :%d",result == TSDK_SUCCESS ?
@"YES" : @"NO",audioRoute);
    return result == TSDK_SUCCESS ? YES : NO;
}
```

步骤2 UI调用tsdk_get_mobile_audio_route()获取移动音频路由设备。

代码示例：

```
-(ROUTE_TYPE)obtainMobileAudioRoute
{
    TSDK_E_MOBILE_AUIDO_ROUTE route;
    TSDK_RESULT result = tsdk_get_mobile_audio_route(&route);
    DDLogInfo(@"tsdk_get_mobile_audio_route result is %d, audioRoute is :%d",result,route);
    return (ROUTE_TYPE)route;
}
```

----结束

管理视频设备

说明

一般用于用户对摄像头进行设置和切换。

步骤1 UI调用tsdk_set_video_orient()设置当前使用的摄像头设备。

代码示例：

```
-(BOOL)switchCameraIndex:(NSUInteger)cameraCaptureIndex callId:(unsigned int)callId
{
    TSDK_S_VIDEO_ORIENT orient;
    memset(&orient, 0, sizeof(TSDK_S_VIDEO_ORIENT));
    orient.choice = 1;
    orient.portrait = 0;
    orient.landscape = 0;
    orient.seascape = 1;
    TSDK_RESULT result = tsdk_set_video_orient(callId, (TSDK_UINT32)cameraCaptureIndex, &orient);
    if (result == TSDK_SUCCESS)
    {
        _cameraCaptureIndex = cameraCaptureIndex == 1 ? CameraIndexFront : CameraIndexBack;
    }
    [self updateVideoRenderInfoWithVideoIndex:(CameraIndex)cameraCaptureIndex
withRenderType:TSDK_E_VIDEO_WND_LOCAL andCallId:callId];
    return result == TSDK_SUCCESS ? YES : NO;
}
```

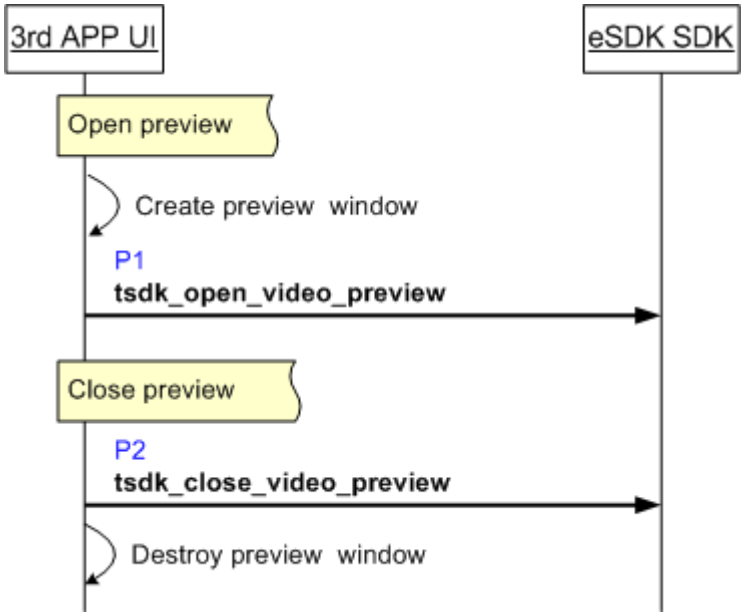
----结束

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 5-13 预览本地视频流程



步骤1 UI先创建本地预览窗口，再调用tsdk_open_video_preview()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。

代码示例：

```
- (BOOL)videoPreview:(unsigned int)cameraIndex toView:(id) viewHandler
{
    _videoPreview = viewHandler;
    TSDK_RESULT ret = tsdk_open_video_preview((TSDK_UPTR)viewHandler, (TSDK_UINT32)cameraIndex);
    DDLogInfo(@"Camera_Log:tsdk_open_video_preview result is %d", ret);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

步骤2 UI调用tsdk_close_video_preview()关闭本地视频预览窗口，同时销毁本地预览窗口。

代码示例：

```
-(void)stopVideoPreview
{
    tsdk_close_video_preview();
}
```

----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。

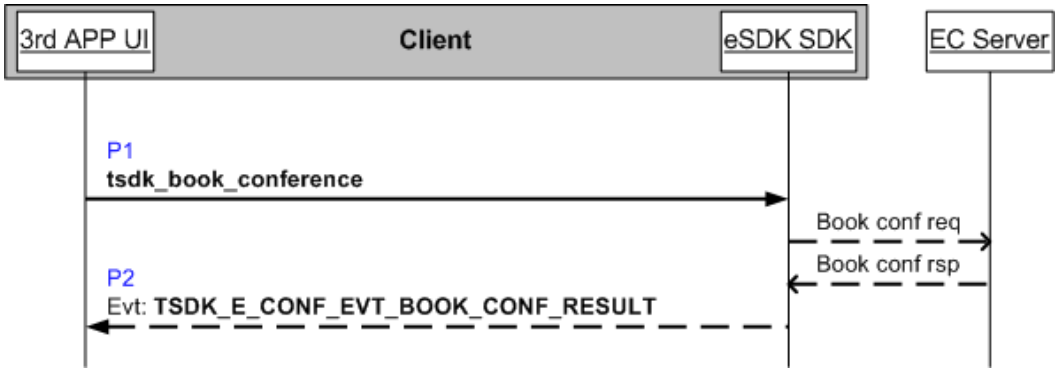
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

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 5-14 预约会议流程



步骤1 UI调用tsdk_book_conference()预约会议，会议参数结构为TSDK_S_BOOK_CONF_INFO，SDK发送预约会议请求至会议服务器。

说明

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_num&attendee_list）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：

```
-(BOOL)tsdkConfctrlBookConf:(NSArray *)attendeeArray mediaType:(EC_CONF_MEDIATYPE)mediaType
startTime:(NSDate *)startTime confLen:(int)confLen subject:(NSString *)subject
{
    TSDK_S_BOOK_CONF_INFO *bookConfInfoUportal = (TSDK_S_BOOK_CONF_INFO
*)malloc(sizeof(TSDK_S_BOOK_CONF_INFO));
    memset_s(bookConfInfoUportal, sizeof(TSDK_S_BOOK_CONF_INFO), 0,
sizeof(TSDK_S_BOOK_CONF_INFO));
    if (subject.length > 0 && subject != nil)
    {
        strcpy(bookConfInfoUportal->subject, [subject UTF8String]);
    }
    bookConfInfoUportal->conf_type = TSDK_E_CONF_INSTANT;
    if (startTime != nil)
    {
        NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
        [dateFormatter setDateFormat:@"yyyy-MM-dd HH:mm"];
        NSString *startTimeStr = [dateFormatter stringFromDate:startTime];
        NSString *utcStr = [self getUTCFormateLocalDate:startTimeStr];
        DDLogInfo(@"start time : %@, utc time: %@",startTimeStr,utcStr);
        strcpy(bookConfInfoUportal->start_time, [utcStr UTF8String]);
        bookConfInfoUportal->duration = confLen;
        bookConfInfoUportal->conf_type = TSDK_E_CONF_RESERVED;
    }
    if (attendeeArray.count == 0)
```

```
    {
        bookConfInfoUportal->size = 5;
        bookConfInfoUportal->attendee_num = 0;
        bookConfInfoUportal->attendee_list = NULL;
    }else{
        bookConfInfoUportal->size = (TSDK_UINT32)attendeeArray.count * 2;
        bookConfInfoUportal->attendee_num = (TSDK_UINT32)attendeeArray.count;
        bookConfInfoUportal->attendee_list = [self returnAttendeeWithArray:attendeeArray];
    }
    bookConfInfoUportal->conf_media_type = (TSDK_E_CONF_MEDIA_TYPE)mediaType;
    bookConfInfoUportal->is_hd_conf = TSDK_FALSE;
    bookConfInfoUportal->is_multi_stream_conf = TSDK_FALSE;
    bookConfInfoUportal->is_auto_record = TSDK_FALSE;
    bookConfInfoUportal->is_auto_prolong = TSDK_TRUE;
    bookConfInfoUportal->is_auto_mute = TSDK_FALSE;
    bookConfInfoUportal->welcome_prompt = TSDK_E_CONF_WARNING_DEFAULT;
    bookConfInfoUportal->enter_prompt = TSDK_E_CONF_WARNING_DEFAULT;
    bookConfInfoUportal->leave_prompt = TSDK_E_CONF_WARNING_DEFAULT;
    bookConfInfoUportal->reminder = TSDK_E_CONF_REMINDER_NONE;
    bookConfInfoUportal->language = TSDK_E_CONF_LANGUAGE_ZH_CN;
    TSDK_RESULT ret = tsdk_book_conference(bookConfInfoUportal);
    DDLogInfo(@"tsdk_book_conference result : %d",ret);
    free(bookConfInfoUportal);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

步骤2 SDK在收到服务器返回的会议预约结果响应后，向UI上报预约会议结果通知TSDK_E_CONF_EVT_BOOK_CONF_RESULT，对应的结果数据结构为TSDK_S_CONF_BASE_INFO。

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：

```
case TSDK_E_CONF_EVT_BOOK_CONF_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_BOOK_CONF_RESULT");
    BOOL result = notify.param1 == TSDK_SUCCESS;
    if (!result)
    {
        DDLogError(@"TSDK_E_CONF_EVT_BOOK_CONF_RESULT,error:%@",[NSString stringWithUTF8String:
(TSDK_CHAR *)notify.data]);
        return;
    }
    TSDK_S_CONF_BASE_INFO *confListInfo = (TSDK_S_CONF_BASE_INFO *)notify.data;
}
```

----结束

注意事项

无。

5.5.2.2 创建即时会议

应用场景

用户创建立即会议。

说明

立即会议创建成功后，用户的SIP号码自动入会，用户作为会议主席召集其他与会者加入会议。

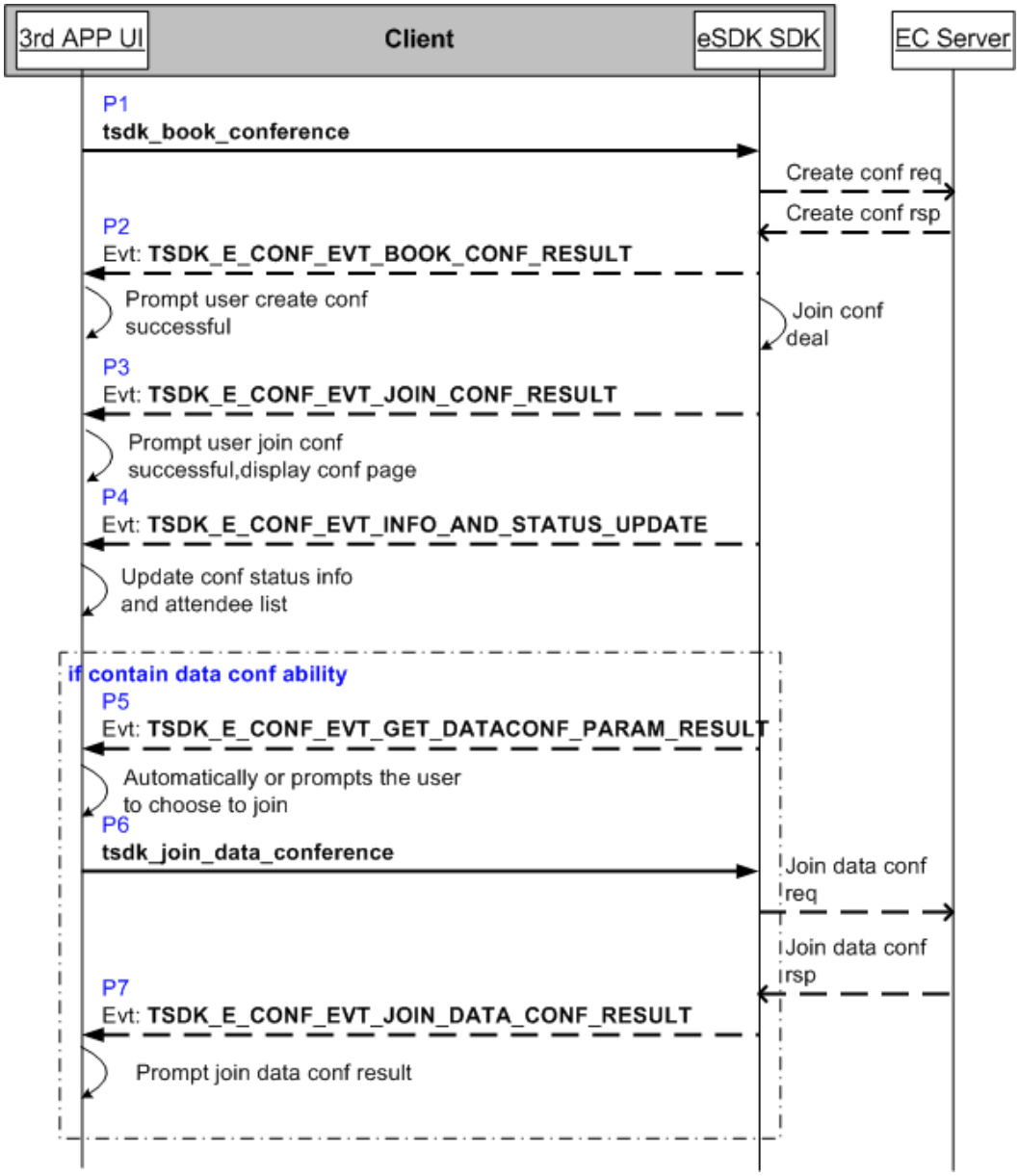
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

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 5-15 创建立即会议流程



步骤1 UI调用tsdk_book_conference()创建立即会议，会议参数结构为TSDK_S_BOOK_CONF_INFO；SDK发送创建会议请求至会议服务器。

说明

在创建即时会议时，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_num&attendee_list）必选，其他参数可选；

1. 预约会议，会议类型应选TSDK_E_CONF_INSTANT。
2. 会议开始时间不用指定。
3. 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：

```
-(BOOL)tsdkConfctrlBookConf:(NSArray *)attendeeArray mediaType:(EC_CONF_MEDIATYPE)mediaType
startTime:(NSDate *)startTime confLen:(int)confLen subject:(NSString *)subject
{
    TSDK_S_BOOK_CONF_INFO *bookConfInfoUportal = (TSDK_S_BOOK_CONF_INFO
*)malloc(sizeof(TSDK_S_BOOK_CONF_INFO));
    memset_s(bookConfInfoUportal, sizeof(TSDK_S_BOOK_CONF_INFO), 0,
sizeof(TSDK_S_BOOK_CONF_INFO));
    if (subject.length > 0 && subject != nil)
    {
        strcpy(bookConfInfoUportal->subject, [subject UTF8String]);
    }
    bookConfInfoUportal->conf_type = TSDK_E_CONF_INSTANT;
    if (startTime != nil)
    {
        NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
        [dateFormatter setDateFormat:@"yyyy-MM-dd HH:mm"];
        NSString *startTimeStr = [dateFormatter stringFromDate:startTime];
        NSString *utcStr = [self getUTCFormateLocalDate:startTimeStr];
        DDLogInfo(@"start time : %@, utc time: %@",startTimeStr,utcStr);
        strcpy(bookConfInfoUportal->start_time, [utcStr UTF8String]);
        bookConfInfoUportal->duration = confLen;
        bookConfInfoUportal->conf_type = TSDK_E_CONF_RESERVED;
    }
    if (attendeeArray.count == 0)
    {
        bookConfInfoUportal->size = 5;
        bookConfInfoUportal->attendee_num = 0;
        bookConfInfoUportal->attendee_list = NULL;
    }else{
        bookConfInfoUportal->size = (TSDK_UINT32)attendeeArray.count * 2;
        bookConfInfoUportal->attendee_num = (TSDK_UINT32)attendeeArray.count;
        bookConfInfoUportal->attendee_list = [self returnAttendeeWithArray:attendeeArray];
    }
    bookConfInfoUportal->conf_media_type = (TSDK_E_CONF_MEDIA_TYPE)mediaType;
    bookConfInfoUportal->is_hd_conf = TSDK_FALSE;
    bookConfInfoUportal->is_multi_stream_conf = TSDK_FALSE;
    bookConfInfoUportal->is_auto_record = TSDK_FALSE;
    bookConfInfoUportal->is_auto_prolong = TSDK_TRUE;
    bookConfInfoUportal->is_auto_mute = TSDK_FALSE;
    bookConfInfoUportal->welcome_prompt = TSDK_E_CONF_WARNING_DEFAULT;
    bookConfInfoUportal->enter_prompt = TSDK_E_CONF_WARNING_DEFAULT;
    bookConfInfoUportal->leave_prompt = TSDK_E_CONF_WARNING_DEFAULT;
    bookConfInfoUportal->reminder = TSDK_E_CONF_REMINDER_NONE;
    bookConfInfoUportal->language = TSDK_E_CONF_LANGUAGE_ZH_CN;
    TSDK_RESULT ret = tsdk_book_conference(bookConfInfoUportal);
    DDLogInfo(@"tsdk_book_conference result : %d",ret);
    free(bookConfInfoUportal);
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

步骤2 SDK在收到服务器返回的立即会议创建响应后，向UI上报会议创建结果通知TSDK_E_CONF_EVT_BOOK_CONF_RESULT，对应的结果数据结构为TSDK_S_CONF_BASE_INFO，UI应提示创建会议成功。

代码示例：

```
case TSDK_E_CONF_EVT_BOOK_CONF_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_BOOK_CONF_RESULT");
    BOOL result = notify.param1 == TSDK_SUCCESS;
    if (!result)
    {
        DDLogError(@"TSDK_E_CONF_EVT_BOOK_CONF_RESULT,error:%@",[NSString stringWithUTF8String:
(TSDK_CHAR *)notify.data]);
        return;
    }
    TSDK_S_CONF_BASE_INFO *confListInfo = (TSDK_S_CONF_BASE_INFO *)notify.data;
}
```


eSDK CloudVC 20.1.RC1 开发指南(iOS)

步骤3 SDK在收到服务器返回的加入会议响应后，向UI上报加入会议结果通知TSDK_E_CONF_EVT_JOIN_CONF_RESULT，对应的结果数据结构为TSDK_S_JOIN_CONF_IND_INFO，并返回conf handle，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，则会向UI上报会议权限请求失败通知TSDK_E_CONF_EVT_REQUEST_CONF_RIGHT_FAILED，UI提示申请会控权限失败，若请求成功，则不上报。

代码示例：

```
case TSDK_E_CONF_EVT_JOIN_CONF_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_JOIN_CONF_RESULT");
    BOOL result = notify.param2 == TSDK_SUCCESS;
    if (!result)
    {
        DDLogError(@"TSDK_E_CONF_EVT_JOIN_CONF_RESULT,error:%@",[NSString stringWithUTF8String:
(TSDK_CHAR *)notify.data]);
        return;
    }
    _confHandle = notify.param1;
    TSDK_S_JOIN_CONF_IND_INFO *confInfo = (TSDK_S_JOIN_CONF_IND_INFO *)notify.data;
    _currentCallId = confInfo->call_id;
    dispatch_async(dispatch_get_main_queue(), ^{
        // go conference
        [self goConferenceRunView:nil];
        [self respondsECConferenceDelegateWithType:CONF_E_CONNECT result:nil];
        [[NSNotificationCenter defaultCenter] postNotificationName:TUP_CALL_REMOVE_CALL_VIEW_NOTIFY
object:nil];
    });
}
break;
```

步骤4 SDK收到会议状态更新通知，向UI上报会议信息及状态更新事件TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE，UI刷新会议成员列表和会议信息。

代码示例：

```
case TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE");
    [self handleAttendeeUpdateNotify:notify];
}
break;
```

步骤5 若会议包含数据会议能力，SDK会向UI上报获取数据会议参数结果TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT。

代码示例：

```
case TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT");
    BOOL result = notify.param2 == TSDK_SUCCESS;
    if (!result)
    {
        DDLogError(@"TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT,error:%@",[NSString
stringWithUTF8String:(TSDK_CHAR *)notify.data]);
        return;
    }
    dispatch_async(dispatch_get_main_queue(), ^{
         [self joinDataConference];
        [self startHeartBeatTimer];
    });
}
break;
```

步骤6 此时UI可选择自动加入或用户选择加入数据会议，调用加入数据会议接口tsdk_join_data_conference()，SDK发送创建会议请求至会议服务器。

代码示例：

```
-(void)joinDataConference
{
    TSDK_RESULT result = tsdk_join_data_conference(_confHandle);
    DDLogInfo(@"tsdk_join_data_conference ret: %d", result);
}
```

步骤7 SDK在收服务器加入数据会议响应后，向UI上报数据会议加入结果通知TSDK_E_CONF_EVT_JOIN_DATA_CONF_RESULT，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，会向UI上报会议信息及状态更新事件TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE，UI刷新会议成员列表和会议信息。

代码示例：

```
case TSDK_E_CONF_EVT_JOIN_DATA_CONF_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_JOIN_DATA_CONF_RESULT");
    NSDictionary *resultInfo = nil;
    BOOL isSuccess = notify.param2 == TSDK_SUCCESS;
    resultInfo = @{
        UCCONF_RESULT_KEY :[NSNumber numberWithBool:isSuccess]
    };
  [self respondsECConferenceDelegateWithType:DATA_CONF_JOIN_RESOULT result:resultInfo];
}
break;
```

----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。

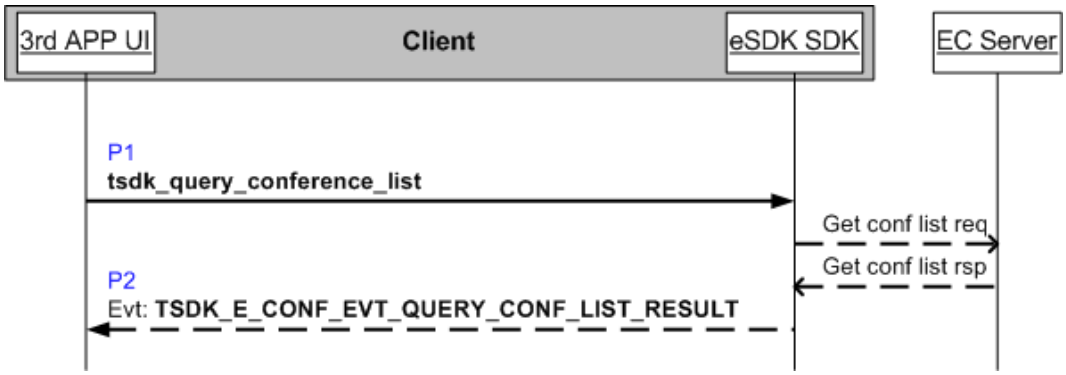
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

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 5-16 查询会议列表流程



步骤1 UI调用tsdk_query_conference_list()查询会议列表，查询会议列表的请求信息结构为TSDK_S_QUERY_CONF_LIST_REQ。

说明

1、会议权限（conf_right）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、请求会议列表页索引（page_index），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、会议列表每页的会议个数（page_size），建议与应用程序会议列表个数相同，必须要有明确值。

4、返回来的时间为UTC时间，UI进行页面呈现之前需要将UTC时间转换为本地时间。

代码示例：

```
-(BOOL)obtainConferenceListWithPageIndex:(int)pageIndex pageSize:(int)pageSize
{
    TSDK_S_QUERY_CONF_LIST_REQ conflistInfo;
    memset(&conflistInfo, 0, sizeof(TSDK_S_QUERY_CONF_LIST_REQ));
    conflistInfo.conf_right = TSDK_E_CONF_RIGHT_CREATE_JOIN;
    conflistInfo.is_include_end = TSDK_FALSE;
    conflistInfo.page_index = pageIndex;
    conflistInfo.page_size = pageSize;
    int result = tsdk_query_conference_list(&conflistInfo);
    DDLogInfo(@"tsdk_query_conference_list result: %d",result);
    return result == TSDK_SUCCESS ? YES : NO;
}
```

步骤2 SDK在收到服务器返回的查询会议列表响应后，向UI上报查询会议列表结果通知TSDK_E_CONF_EVT_QUERY_CONF_LIST_RESULT，对应的结果数据结构为TSDK_S_CONF_LIST_INFO。

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：

```
case TSDK_E_CONF_EVT_QUERY_CONF_LIST_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_QUERY_CONF_LIST_RESULT");
    BOOL result = notify.param1 == TSDK_SUCCESS;
    if (!result)
```



```
    {
        DDLogError(@"TSDK_E_CONF_EVT_QUERY_CONF_LIST_RESULT,error:%@",[NSString
stringWithUTF8String:(TSDK_CHAR *)notify.data]);
        return;
    }
    [self handleGetConfListResult:notify];
}
break;
```

----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

用户通过会议列表一键入会的方式加入会议。

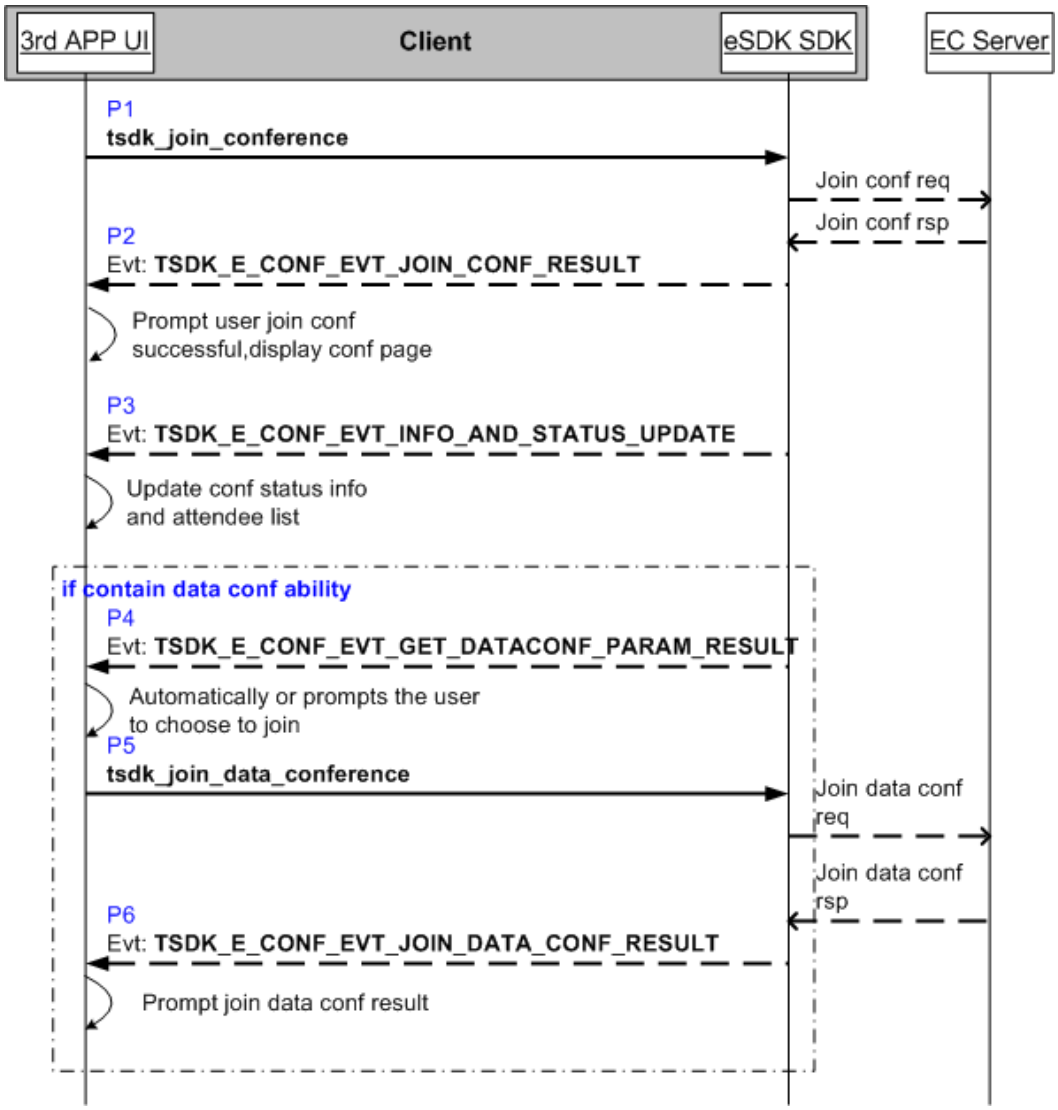
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

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 5-17 会议列表一键入会流程



步骤1 UI调用tsdk_join_conference()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，入会参数（conf_join_param），是否视频接入会议（is_video_join）为必选；

1. 入会参数结构体中conf_id，access_number，conf_password都为必选。
2. 入会号码（join_number）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call_id,需要记录下），在使用SIP终端号码入会时有效。

代码示例：

```
-(BOOL)joinConferenceWithConfId:(NSString *)confId AccessNumber:(NSString *)accessNumber
confPassWord:(NSString *)confPassWord joinNumber:(NSString *)joinNumber isVideoJoin:(BOOL)isVideoJoin
{
    TSDK_S_CONF_JOIN_PARAM confJoinParam;
    memset(&confJoinParam, 0, sizeof(TSDK_S_CONF_JOIN_PARAM));
    if (confId.length > 0 && confId != nil)
```

```
    {
        strcpy(confJoinParam.conf_id, [confId UTF8String]);
    }
    if (confPassWord.length > 0 && confPassWord != nil)
    {
        strcpy(confJoinParam.conf_password, [confPassWord UTF8String]);
    }
    if (accessNumber.length > 0 && accessNumber != nil)
    {
        strcpy(confJoinParam.access_number, [accessNumber UTF8String]);
    }
    TSDK_CHAR join_number;
    if (!self.selfJoinNumber)
    {
        self.selfJoinNumber = self.sipAccount;
    }
    strcpy(&join_number, [self.selfJoinNumber UTF8String]);
    TSDK_UINT32 call_id;
    BOOL result = tsdk_join_conference(&confJoinParam, &join_number, (TSDK_BOOL)isVideoJoin,
&call_id);
    DDLogInfo(@"tsdk_join_conference = %d, call_id is :%d",result,call_id);
    return result == TSDK_SUCCESS ? YES : NO;
}
```

步骤2 SDK在收到服务器返回的加入会议响应后，向UI上报加入会议结果通知TSDK_E_CONF_EVT_JOIN_CONF_RESULT，对应的结果数据结构为TSDK_S_JOIN_CONF_IND_INFO，并返回conf handle，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，则会向UI上报会议权限请求失败通知TSDK_E_CONF_EVT_REQUEST_CONF_RIGHT_FAILED，UI提示申请会控权限失败，若请求成功，则不上报。

代码示例：

```
case TSDK_E_CONF_EVT_JOIN_CONF_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_JOIN_CONF_RESULT");
    BOOL result = notify.param2 == TSDK_SUCCESS;
    if (!result)
    {
        DDLogError(@"TSDK_E_CONF_EVT_JOIN_CONF_RESULT,error:%@",[NSString stringWithUTF8String:
(TSDK_CHAR *)notify.data]);
        return;
    }
    _confHandle = notify.param1;
    TSDK_S_JOIN_CONF_IND_INFO *confInfo = (TSDK_S_JOIN_CONF_IND_INFO *)notify.data;
    _currentCallId = confInfo->call_id;
    dispatch_async(dispatch_get_main_queue(), ^{
        // go conference
        [self goConferenceRunView:nil];
        [self respondsECConferenceDelegateWithType:CONF_E_CONNECT result:nil];
        [[NSNotificationCenter defaultCenter] postNotificationName:TUP_CALL_REMOVE_CALL_VIEW_NOTIFY
object:nil];
    });
}
break;
```

步骤3 SDK收到会议状态更新通知，向UI上报会议信息及状态更新事件TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE，UI刷新会议成员列表和会议信息。

代码示例：

```
case TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE");
```

```
    [self handleAttendeeUpdateNotify:notify];
}
break;
```

步骤4 若会议包含数据会议能力，SDK会向UI上报获取数据会议参数结果TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT。

代码示例：

```
case TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT");
    BOOL result = notify.param2 == TSDK_SUCCESS;
    if (!result)
    {
        DDLogError(@"TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT,error:%@",[NSString
stringWithUTF8String:(TSDK_CHAR *)notify.data]);
        return;
    }
    dispatch_async(dispatch_get_main_queue(), ^{
        [self joinDataConference];
        [self startHeartBeatTimer];
    });
}
break;
```

步骤5 此时UI可选择自动加入或用户选择加入数据会议，调用加入数据会议接口tsdk_join_data_conference()，SDK发送创建会议请求至会议服务器。

代码示例：

```
-(void)joinDataConference
{
    TSDK_RESULT result = tsdk_join_data_conference(_confHandle);
    DDLogInfo(@"tsdk_join_data_conference ret: %d", result);
}
```

步骤6 SDK在收服务器加入数据会议响应后，向UI上报数据会议加入结果通知TSDK_E_CONF_EVT_JOIN_DATA_CONF_RESULT，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，会向UI上报会议信息及状态更新事件TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE，UI刷新会议成员列表和会议信息。

代码示例：

```
case TSDK_E_CONF_EVT_JOIN_DATA_CONF_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_JOIN_DATA_CONF_RESULT");
    NSDictionary *resultInfo = nil;
    BOOL isSuccess = notify.param2 == TSDK_SUCCESS;
    resultInfo = @{
        UCCONF_RESULT_KEY :[NSNumber numberWithBool:isSuccess]
    };
  [self respondsECConferenceDelegateWithType:DATA_CONF_JOIN_RESOULT result:resultInfo];
}
break;
```

----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。

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。

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

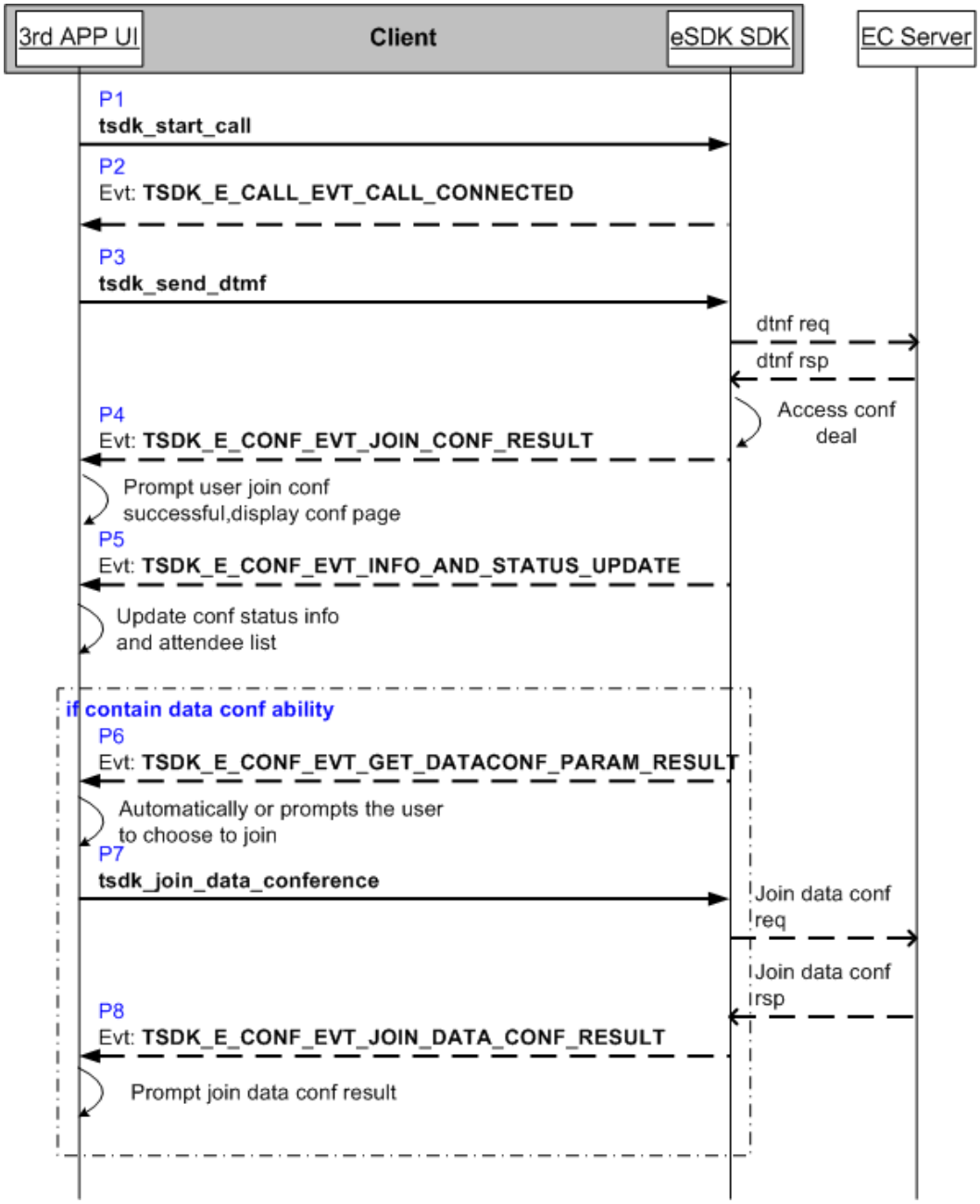
前提条件

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 5-18 统一会议接入号入会流程



步骤1 UI调用“音视频呼叫”接口tsdk_start_call()发起一个呼叫请求，被叫号码为“统一会议接入号”。

代码示例：

```
TSDK_BOOL isVideo = ((TSDK_CALL_E_CALL_TYPE)callType==CALL_VIDEO)?TSDK_TRUE:TSDK_FALSE;
TSDK_UINT32 callid = 0;
TSDK_RESULT ret = tsdk_start_call(&callid,(TSDK_CHAR*)[number UTF8String], isVideo);
```

eSDK CloudVC 20.1.RC1 开发指南(iOS)

步骤2 若呼叫建立 成功，SDK向UI上报呼叫建立事件TSDK_E_CALL_EVT_CALL_CONNECTED，与普通音视频呼叫相同。

代码示例：

```
case TSDK_E_CALL_EVT_CALL_CONNECTED:
{
    DDLogInfo(@"Call_Log: recv call notify :CALL_E_EVT_CALL_CONNECTED");
    TSDK_S_CALL_INFO *callInfo = (TSDK_S_CALL_INFO *)notify.data;
    CallInfo *tsdkCallInfo = [CallInfo transfromFromCallInfoStract:callInfo];
    NSString *callId = [NSString stringWithFormat:@"%d", tsdkCallInfo.stateInfo.callId];
    [_tsdkCallInfoDic setObject:tsdkCallInfo forKey:callId];
    NSDictionary *resultInfo = @{
        TSDK_CALL_INFO_KEY : tsdkCallInfo
    };
    [self respondsCallDelegateWithType:CALL_CONNECT result:resultInfo];
}
```

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，UI调用tsdk_send_dtmf()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：

```
- (BOOL)sendDTMFWithDialNum:(NSString *)number callId:(unsigned int)callId
{
    TSDK_E_DTMF_TONE dtmfTone = (TSDK_E_DTMF_TONE)[number intValue];
    if ([number isEqualToString:@"*"])
    {
        dtmfTone = TSDK_E_DTMF_STAR;
    }
    else if ([number isEqualToString:@"#"])
    {
        dtmfTone = TSDK_E_DTMF_POUND;
    }
    TSDK_UINT32 callid = callId;
    TSDK_RESULT ret = tsdk_send_dtmf((TSDK_UINT32)callid,(TSDK_E_DTMF_TONE)dtmfTone);
    DDLogInfo(@"Call_Log: tsdk_send_dtmf = %@",(TSDK_SUCCESS == ret)?@"YES":@"NO");
    return ret == TSDK_SUCCESS ? YES : NO;
}
```

步骤4 SDK在收到服务器返回的加入会议响应后，向UI上报加入会议结果通知TSDK_E_CONF_EVT_JOIN_CONF_RESULT，对应的结果数据结构为TSDK_S_JOIN_CONF_IND_INFO，并返回conf handle，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，则会向UI上报会议权限请求失败通知TSDK_E_CONF_EVT_REQUEST_CONF_RIGHT_FAILED，UI提示申请会控权限失败，若请求成功，则不上报。

代码示例：

```
case TSDK_E_CONF_EVT_JOIN_CONF_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_JOIN_CONF_RESULT");
    BOOL result = notify.param2 == TSDK_SUCCESS;
    if (!result)
    {
        DDLogError(@"TSDK_E_CONF_EVT_JOIN_CONF_RESULT,error:%@",[NSString stringWithUTF8String:
(TSDK_CHAR *)notify.data]);
        return;
    }
    _confHandle = notify.param1;
    TSDK_S_JOIN_CONF_IND_INFO *confInfo = (TSDK_S_JOIN_CONF_IND_INFO *)notify.data;
    _currentCallId = confInfo->call_id;
    dispatch_async(dispatch_get_main_queue(), ^{
        // go conference
        [self goConferenceRunView:nil];
        [self respondsECConferenceDelegateWithType:CONF_E_CONNECT result:nil];
```

```
        [[NSNotificationCenter defaultCenter] postNotificationName:TUP_CALL_REMOVE_CALL_VIEW_NOTIFY
object:nil];
    });
}
break;
```

步骤5 SDK收到会议状态更新通知，向UI上报会议信息及状态更新事件TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE，UI刷新会议成员列表和会议信息。

代码示例：

```
case TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE");
    [self handleAttendeeUpdateNotify:notify];
}
break;
```

步骤6 若会议包含数据会议能力，SDK会向UI上报获取数据会议参数结果TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT。

代码示例：

```
case TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT");
    BOOL result = notify.param2 == TSDK_SUCCESS;
    if (!result)
    {
        DDLogError(@"TSDK_E_CONF_EVT_GET_DATACONF_PARAM_RESULT,error:%@",[NSString
stringWithUTF8String:(TSDK_CHAR *)notify.data]);
        return;
    }
    dispatch_async(dispatch_get_main_queue(), ^{
        [self joinDataConference];
        [self startHeartBeatTimer];
    });
}
break;
```

步骤7 此时UI可选择自动加入或用户选择加入数据会议，调用加入数据会议接口tsdk_join_data_conference，SDK发送创建会议请求至会议服务器。

代码示例：

```
-(void)joinDataConference
{
    TSDK_RESULT result = tsdk_join_data_conference(_confHandle);
    DDLogInfo(@"tsdk_join_data_conference ret: %d", result);
}
```

步骤8 SDK在收服务器加入数据会议响应后，向UI上报数据会议加入结果通知TSDK_E_CONF_EVT_JOIN_DATA_CONF_RESULT，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，会向UI上报会议信息及状态更新事件TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE，UI刷新会议成员列表和会议信息。

代码示例：

```
case TSDK_E_CONF_EVT_JOIN_DATA_CONF_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_JOIN_DATA_CONF_RESULT");
    NSDictionary *resultInfo = nil;
    BOOL isSuccess = notify.param2 == TSDK_SUCCESS;
    resultInfo = @{
        UCCONF_RESULT_KEY :[NSNumber numberWithBool:isSuccess]
    };
```



```
  [self respondsECConferenceDelegateWithType:DATA_CONF_JOIN_RESOULT result:resultInfo];
}
break;
```

----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。

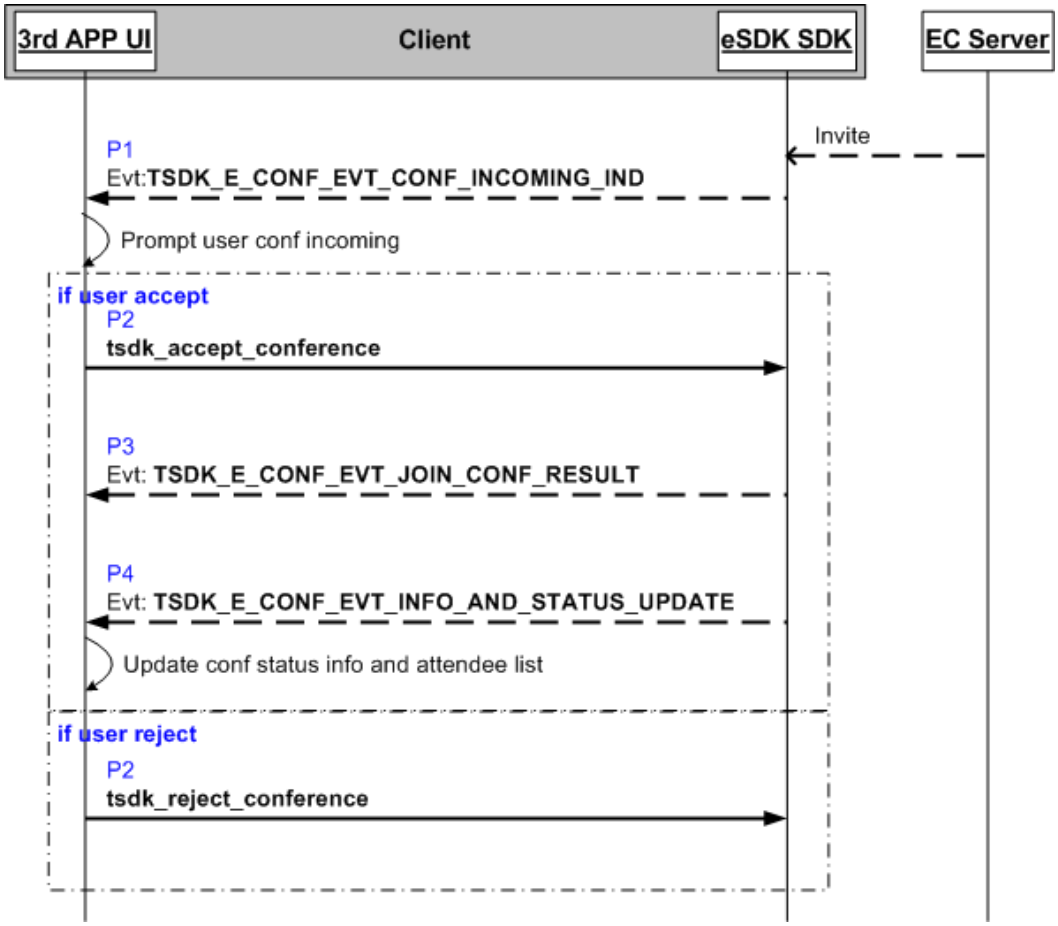
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

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 5-19 被邀入会流程



步骤1 被邀与会者侧SDK收到会议来电请求，向UI上报会议来电事件TSDK_E_CONF_EVT_CONF_INCOMING_IND，对应的事件数据结构为TSDK_S_CONF_INCOMING_INFO，UI提示用户会议来电。

代码示例：

```
case TSDK_E_CONF_EVT_CONF_INCOMING_IND:
{
    if (!self.selfJoinNumber)
    {
        self.selfJoinNumber = self.sipAccount;
    }
    DDLogInfo(@"TSDK_E_CONF_EVT_CONF_INCOMING_IND");
    int callID = notify.param2;
    _confHandle = notify.param1;
    TSDK_S_CONF_INCOMING_INFO *inComingInfo = (TSDK_S_CONF_INCOMING_INFO
*)notify.data;
    CallInfo *tsdkCallInfo = [[CallInfo alloc]init];
    tsdkCallInfo.stateInfo.callId = callID;
    BOOL is_video_conf = NO;
    if (inComingInfo->conf_media_type == TSDK_E_CONF_MEDIA_VIDEO || inComingInfo->conf_media_type
== TSDK_E_CONF_MEDIA_VIDEO_DATA)
    {
        is_video_conf = YES;
    }
    tsdkCallInfo.stateInfo.callType = is_video_conf?CALL_VIDEO:CALL_AUDIO;
```

```
    tsdkCallInfo.stateInfo.callNum = [NSString stringWithUTF8String:inComingInfo->number];
    [[NSNotificationCenter defaultCenter] postNotificationName:EC_COMING_CONF_NOTIFY
object:nil                                                    userInfo:@{TUP_CONF_INCOMING_KEY :
tsdkCallInfo}];
}
break;
```

步骤2 被邀与会者侧接受入会邀请则调用tsdk_accept_conference()接听会议来电呼叫，拒绝入会邀请则调用tsdk_reject_conference()。

代码示例：

```
- (BOOL)acceptConfCallIsJoinVideoConf:(BOOL)isJoinVideoConf
{
    BOOL result = tsdk_accept_conference(_confHandle, isJoinVideoConf);
    DDLogInfo(@"tsdk_accept_conference = %d, _confHandle is :%d",result,_confHandle);
    return result == TSDK_SUCCESS ? YES : NO;
}
```

步骤3 SDK在收到服务器返回的加入会议响应后，向UI上报加入会议结果通知TSDK_E_CONF_EVT_JOIN_CONF_RESULT，对应的结果数据结构为TSDK_S_JOIN_CONF_IND_INFO，并返回conf handle，后续会控时使用，此时， UI可跳转至会议界面。

代码示例：

```
case TSDK_E_CONF_EVT_JOIN_CONF_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_JOIN_CONF_RESULT");
    BOOL result = notify.param2 == TSDK_SUCCESS;
    if (!result)
    {
        DDLogError(@"TSDK_E_CONF_EVT_JOIN_CONF_RESULT,error:%@",[NSString stringWithUTF8String:
(TSDK_CHAR *)notify.data]);
        return;
    }
    _confHandle = notify.param1;
    TSDK_S_JOIN_CONF_IND_INFO *confInfo = (TSDK_S_JOIN_CONF_IND_INFO *)notify.data;
    _currentCallId = confInfo->call_id;
    dispatch_async(dispatch_get_main_queue(), ^{
        // go conference
        [self goConferenceRunView:nil];
        [self respondsECConferenceDelegateWithType:CONF_E_CONNECT result:nil];
        [[NSNotificationCenter defaultCenter] postNotificationName:TUP_CALL_REMOVE_CALL_VIEW_NOTIFY
object:nil];
    });
}
break;
```

步骤4 SDK收到会议成员列表刷新通知，向UI上报会议成员列表刷新事件TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE，UI刷新会议成员列表和会议信息。

代码示例：

```
case TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE");
    [self handleAttendeeUpdateNotify:notify];
}
break;
```

----结束

5.5.3.5 匿名加入会议

应用场景

用户在未注册VC账号时，通过匿名方式加入一个会议。

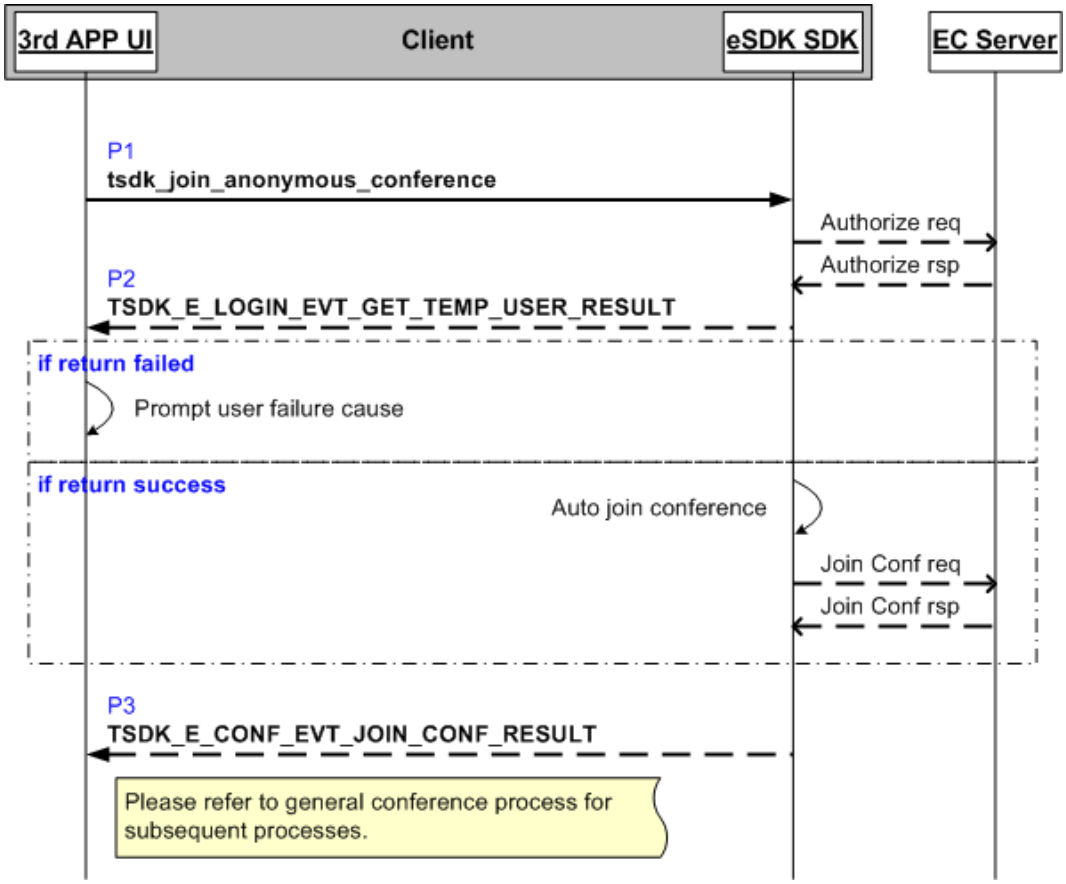
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

会议已经创建，且用户通过第三方方式获取到会议地址和随机码。

流程说明

图 5-20 匿名加入会议流程



步骤1 UI调用tsdk_join_anonymous_conference()接口，匿名加入会议。

说明

参数TSDK_S_CONF_ANONYMOUS_JOIN_PARAM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：

```
- (BOOL)joinConferenceWithDisPlayName:(NSString *)disPlayName ConfId:(NSString *)confID PassWord:(NSString *)passWord ServerAdd:(NSString *)serverAdd ServerPort:(int)serverPort
{
    TSDK_S_CONF_ANONYMOUS_JOIN_PARAM anonymousParam;
    memset(&anonymousParam, 0, sizeof(TSDK_S_CONF_ANONYMOUS_JOIN_PARAM));
    strcpy(anonymousParam.display_name, [disPlayName UTF8String]);
    strcpy(anonymousParam.conf_id, [confID UTF8String]);
    strcpy(anonymousParam.conf_password, [passWord UTF8String]);
    strcpy(anonymousParam.server_addr, [serverAdd UTF8String]);
    anonymousParam.server_port = serverPort;
    TSDK_RESULT joinConfResult = tsdk_join_conference_by_anonymous(&anonymousParam);
    return joinConfResult == TSDK_SUCCESS;
}
```

步骤2 SDK通过会议ID和密码向会议服务器完成鉴权，获取临时账号，上报临时账号获取结果事件TSDK_E_LOGIN_EVT_GET_TEMP_USER_RESULT。

说明

若此次事件通知返回失败，应用程序界面应提示用户。

步骤3 SDK自动完成临时账号注册，并完成加入会议处理，向UI上报加入会议结果事件TSDK_E_CONF_EVT_JOIN_CONF_RESULT。后续过程与普通入会流程相同。

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：

```
case TSDK_E_CONF_EVT_JOIN_CONF_RESULT:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_JOIN_CONF_RESULT");
    BOOL result = notify.param2 == TSDK_SUCCESS;
    if (!result)
    {
        DDLogError(@"TSDK_E_CONF_EVT_JOIN_CONF_RESULT,error:%@",
        [NSString stringWithUTF8String:(TSDK_CHAR *)notify.data]);
        return;
     }
    _confHandle = notify.param1;
    TSDK_S_JOIN_CONF_IND_INFO *confInfo = (TSDK_S_JOIN_CONF_IND_INFO *)notify.data;
    _currentCallId = confInfo->call_id;
    dispatch_async(dispatch_get_main_queue(), ^{
        // go conference
        DDLogInfo(@"goConferenceRunView");
        [self respondsECConferenceDelegateWithType:CONF_E_CONNECT result:nil];
        [[NSNotificationCenter defaultCenter] postNotificationName:TUP_CALL_REMOVE_CALL_VIEW_NOTIFY object:nil];
    });
}
break;
```

----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。

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

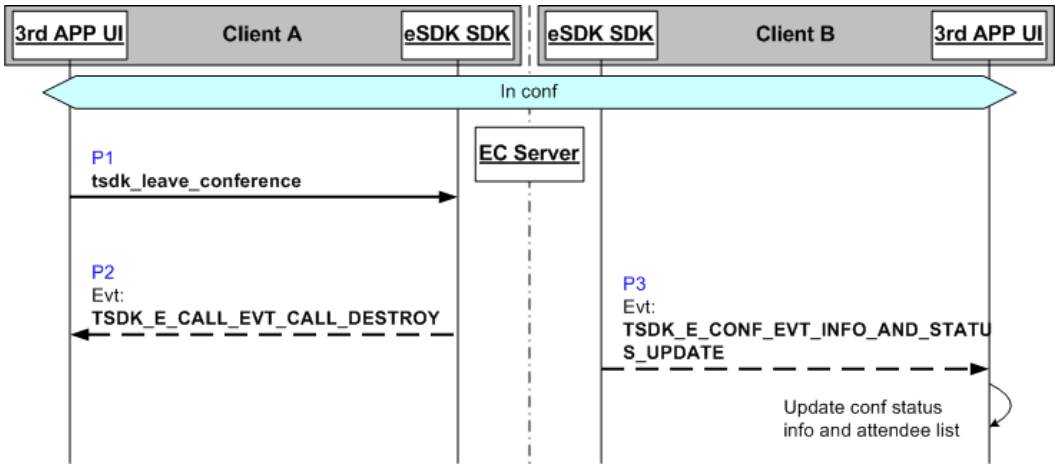
前提条件

与会者已在会议中。

流程说明

退出会议

图 5-21 退出会议流程



步骤1 普通与会者或主席侧UI调用tsdk_leave_conference()主动退出会议。

说明

在主动离开会议过程中，sdk层会主动挂断通话，然后才离开会议。

代码示例：

```
-(BOOL)confCtrlLeaveConference
{
    int result = tsdk_leave_conference(_confHandle);
```

```
    return result == TSDK_SUCCESS ? YES : NO;
}
```

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，向UI上报删除呼叫id事件TSDK_E_CALL_EVT_CALL_DESTROY。

代码示例：

```
case TSDK_E_CALL_EVT_CALL_DESTROY:
{
    [self respondsCallDelegateWithType:CALL_DESTROY result:nil];
}
```

步骤3 其他与会者侧SDK收到会议成员列表刷新通知，向UI上报会议成员列表刷新事件TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE，UI刷新会议成员列表。

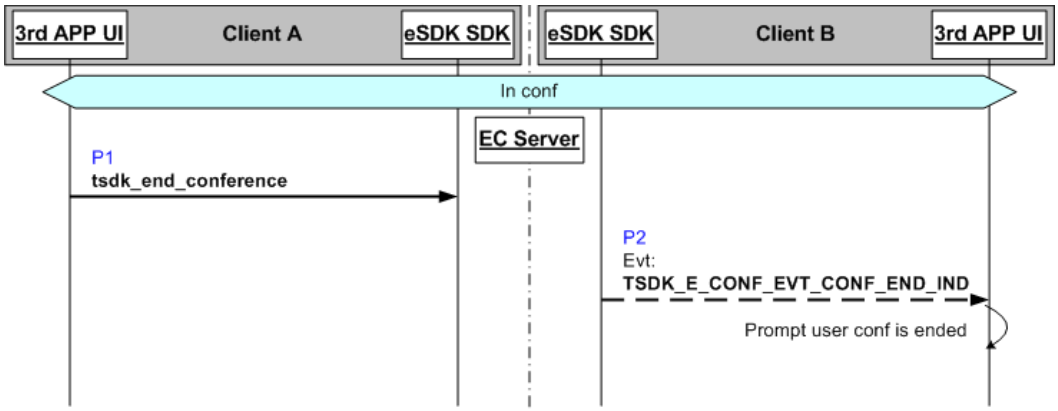
说明

详细流程参见“**更新会议状态信息和与会者列表**”描述。

----结束

结束会议

图 5-22 结束会议流程



步骤1 主席侧UI调用tsdk_end_conference()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：

```
-(BOOL)confCtrlEndConference
{
    int result = tsdk_end_conference(_confHandle);
    return result == TSDK_SUCCESS ? YES : NO;
}
```

步骤2 其他与会者侧SDK收到会议结束通知，向UI上报会议结束事件TSDK_E_CONF_EVT_CONF_END_IND，UI提示用户会议结束。

----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。

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

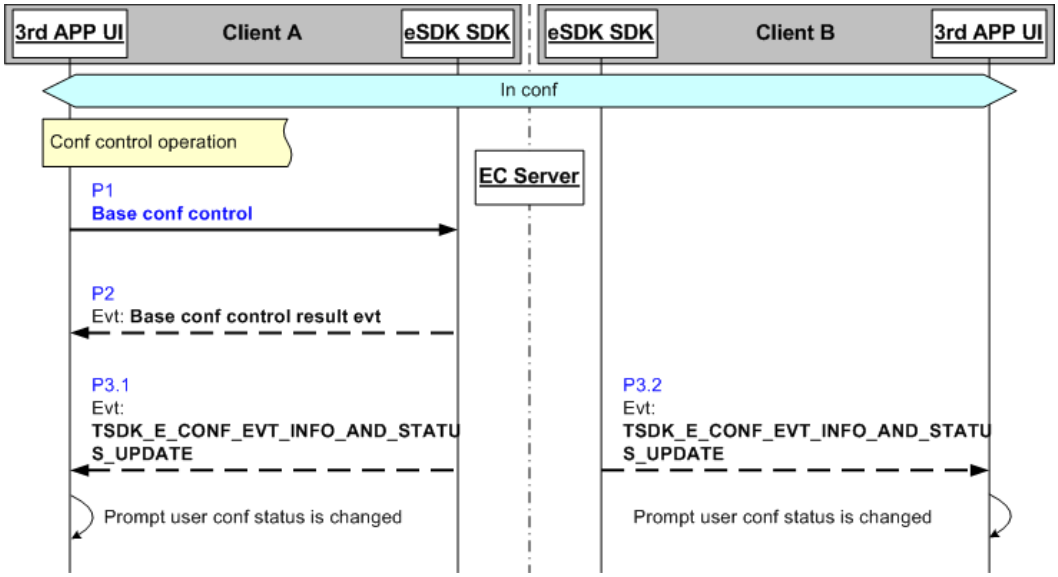
前提条件

无。

流程说明

基础会议控制

图 5-23 基础会议控制流程



步骤1 UI调用会议基础控制接口（如表1），实现会议控制相关操作。

表 5-1

会控类型	接口	权限	说明
闭音会场	tsdk_mute_conference	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）

会控类型	接口	权限	说明
锁定会议	tsdk_lock_conference	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	tsdk_add_attendee	主席	支持邀请一个或多个与会者
重拨与会者	tsdk_redial_attendee	主席	
挂断与会者	tsdk_hang_up_attendee	主席	挂断在会议中的与会者
删除与会者	tsdk_remove_attendee	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
闭音与会者	tsdk_mute_attendee	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	tsdk_set_handup	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
观看与会者	tsdk_watch_attendee	主席 普通与会者	
广播与会者	tsdk_broadcast_attendee	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	tsdk_request_chairman	普通与会者	
释放主席	tsdk_release_chairman	主席	
延长会议	tsdk_postpone_conference	主席	
设置主讲人	tsdk_set_presenter	主席 会议主讲人	会议类型为TSDK_E_CONF_MEDIA_VOICE_DATA或TSDK_E_CONF_MEDIA_VIDEO_DATA支持
申请主讲人	tsdk_request_presenter	与会者	会议类型为TSDK_E_CONF_MEDIA_VOICE_DATA或TSDK_E_CONF_MEDIA_VIDEO_DATA支持

代码示例：

```
-(BOOL)confCtrlMuteConference:(BOOL)isMute
{
    TSDK_BOOL tupBool = isMute ? TSDK_TRUE : TSDK_FALSE;
    int result = tsdk_mute_conference(_confHandle, tupBool);
    return result == TSDK_SUCCESS ? YES : NO;
}
```

步骤2 SDK在收到会控请求的响应后，向UI上报会控操作结果事件TSDK_E_CONF_EVT_CONFCTRL_OPERATION_RESULT。

步骤3 主席侧和其他与会者侧SDK收到会议成员列表刷新通知，向UI上报会议成员列表刷新事件TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE，UI刷新会议页面。

说明

详细流程参见“**更新会议状态信息和与会者列表**”描述。

----结束

注意事项

无。

5.5.4.3 更新会议状态信息和与会者列表

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。

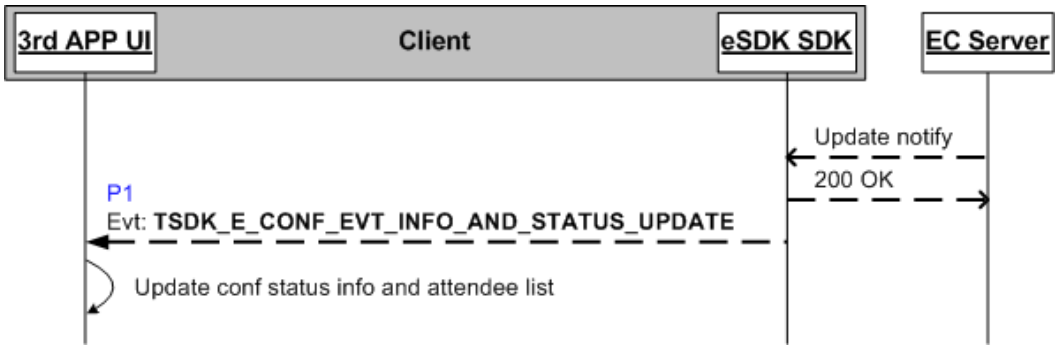
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

主席和与会者均已在会议中。

流程说明

图 5-24 更新会议状态信息和与会者列表流程



步骤1 SDK在收到服务器的与会者列表更新通知后，向UI上报与会者列表更新事件TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE。

说明

事件对应的数据结构TSDK_S_CONF_STATUS_INFO中：

- is_record，表示会场录音状态，为true时，应用程序界面应提示与会者当前处于录音状态；
- is_lock，表示会场锁定状态，为true时，应用程序界面应提示与会者当前处于锁定状态；
- is_all_mute，表示会场静音状态，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；
- subject，会议主题，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；
- is_hd_conf，高清视频会议，为true时，会议视频为高清视频；
- conf_media_type，会议媒体类型，参考结构体TSDK_E_CONF_MEDIA_TYPE；
- conf_state，会议状态，参考结构体TSDK_E_CONF_STATE；
- update_type，表示成员更新方式，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；
- attendee_num，当前会议中的与会者个数；
- attendee_list，与会者的详细信息，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；
- is_live_broadcast，表示会场直播状态，为true时，应用程序应提示与会者当前出于直播状态；
- is_support_live_broadcast，表示会场是否支持直播；
- is_support_record_broadcast，表示会场是否支持录播.

代码示例：

```
case TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_INFO_AND_STATUS_UPDATE");
    [self handleAttendeeUpdateNotify:notify];
}
break;
```

----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。

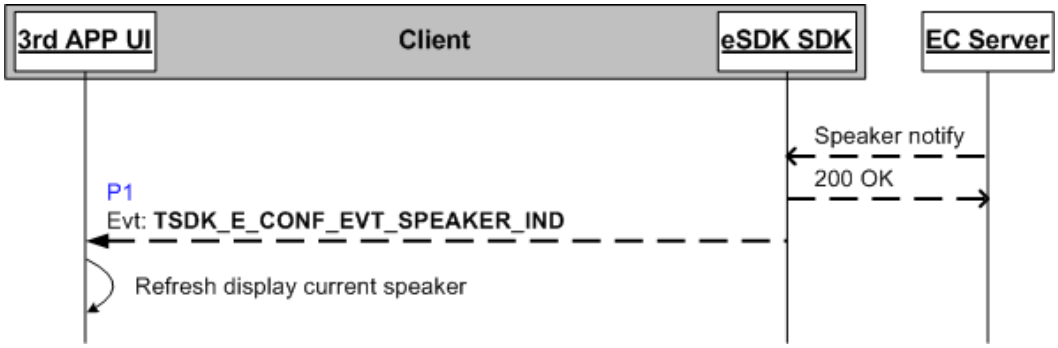
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

与会者均已在会议中。

流程说明

图 5-25 显示发言人流程



步骤1 SDK在收到服务器的发言人通知后，向UI上报发言人通知事件TSDK_E_CONF_EVT_SPEAKER_IND，携带当前发言人的数量、发言人号码和发言音量信息。

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。

代码示例：

```
None
```

----结束

注意事项

无。

5.5.5 桌面协同与共享

5.5.5.1 屏幕共享

应用场景

会议中，移动与会者观看屏幕共享。

说明

移动应用程序暂不具备共享屏幕的能力，屏幕或程序的共享者为PC应用程序。

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

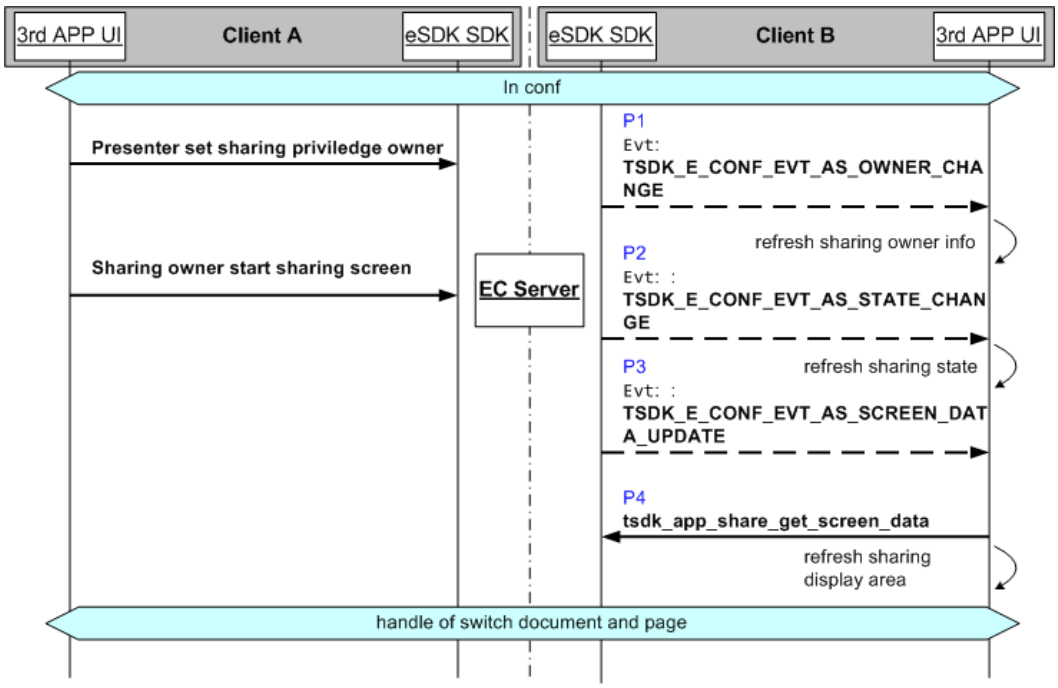
1、加入数据会议成功；

2、加载屏幕共享模块成功。

流程说明

开始观看屏幕或程序共享

图 5-26 开始观看屏幕共享流程



步骤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，移动与会者侧SDK向UI上报共享者变更消息，对应的消息ID为

TSDK_E_CONF_EVT_AS_OWNER_CHANGE，携带当前共享权限拥有者ID，UI刷新共享者信息

步骤2 共享者开始共享屏幕或程序，移动与会者侧SDK上报屏幕共享状态变更通知消息，对应的消息ID为TSDK_E_CONF_EVT_AS_STATE_CHANGE，UI刷新屏幕共享状态信息。

代码示例：

```
case TSDK_E_CONF_EVT_AS_STATE_CHANGE:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_AS_STATE_CHANGE");
    TSDK_S_CONF_AS_STATE_INFO *shareState = (TSDK_S_CONF_AS_STATE_INFO
*)notify.data;                    TSDK_E_CONF_SHARE_STATE state =  shareState->state;
    if (state == TSDK_E_CONF_AS_STATE_NULL)
    {
        [self respondsECConferenceDelegateWithType:DATACONF_SHARE_SCREEN_DATA_STOP
result:nil];
    }
}
break;
```

步骤3 共享侧SDK自动抓取屏幕数据，由业务服务器发送给其他与会者，移动与会者侧SDK向UI上报屏幕数据更新通知消息，对应的消息ID为TSDK_E_CONF_EVT_AS_SCREEN_DATA_UPDATE。

代码示例：

```
case TSDK_E_CONF_EVT_AS_SCREEN_DATA_UPDATE:
{
    DDLogInfo(@"TSDK_E_CONF_EVT_AS_SCREEN_DATA_UPDATE");
    [self handleScreenShareDataConfhandle:notify.param1];
}
break;
```

步骤4 UI调用tsdk_app_share_get_screen_data()接口获取屏幕数据，刷新共享显示区域。

说明

屏幕共享数据通过接口同步返回，转换成图片格式在UI显示。

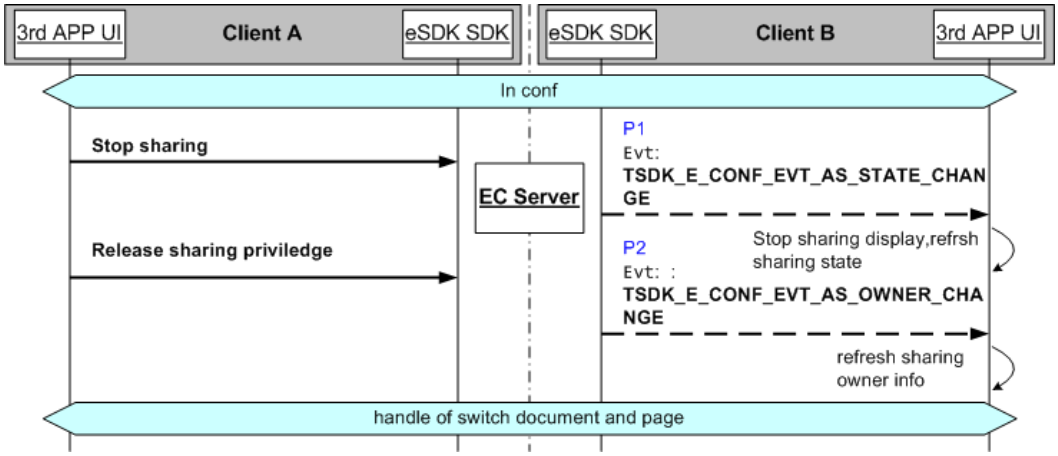
代码示例：

```
-(void)handleScreenShareDataConfhandle:(TSDK_UINT32 )confHandle
{
    TSDK_S_CONF_AS_SCREEN_DATA screenData;
    memset((void *)(&screenData), 0, sizeof(screenData));
// get data info
    TSDK_RESULT dataRet = tsdk_app_share_get_screen_data(confHandle, &screenData);
    if (dataRet != TSDK_SUCCESS)
    {
        DDLogInfo(@"tsdk_app_share_get_screen_data failed:%d",dataRet);
        return;
    }
    DDLogInfo(@"tsdk_app_share_get_screen_data :%d",dataRet);
    char *data = (char *)screenData.data;
    TSDK_UINT32 ssize = *((TSDK_UINT32 *)((char *)data + sizeof(TSDK_UINT16)));
    NSData *imageData = [NSData dataWithBytes:data length:ssize];
    UIImage *image = [[UIImage alloc] initWithData:imageData];
    if (image == nil)
    {
        DDLogInfo(@"share image from data fail!");
        return;
    }
    NSDictionary *shareDataInfo = @{
        DATACONF_SHARE_DATA_KEY:image
    };
    [self respondsECConferenceDelegateWithType:DATA_CONF_AS_ON_SCREEN_DATA result:shareDataInfo];
}
```

----结束

屏幕共享结束处理

图 5-27 屏幕或程序共享结束处理流程



说明

当前图示为共享者主动停止共享，若主讲人结束共享者共享，则对于观看侧，相应的消息顺序相反。

步骤1 共享者停止共享，移动与会者侧SDK通过上报屏幕共享状态变更通知消息，对应的消息ID为TSDK_E_CONF_EVT_AS_STATE_CHANGE，UI刷新屏幕共享状态信息。

步骤2 共享者释放共享权限，移动与会者侧SDK向UI上报共享者变更消息，对应的消息ID为TSDK_E_CONF_EVT_AS_OWNER_CHANGE，UI刷新屏幕共享状态信息。

----结束

注意事项

无。

5.5.5.2 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。

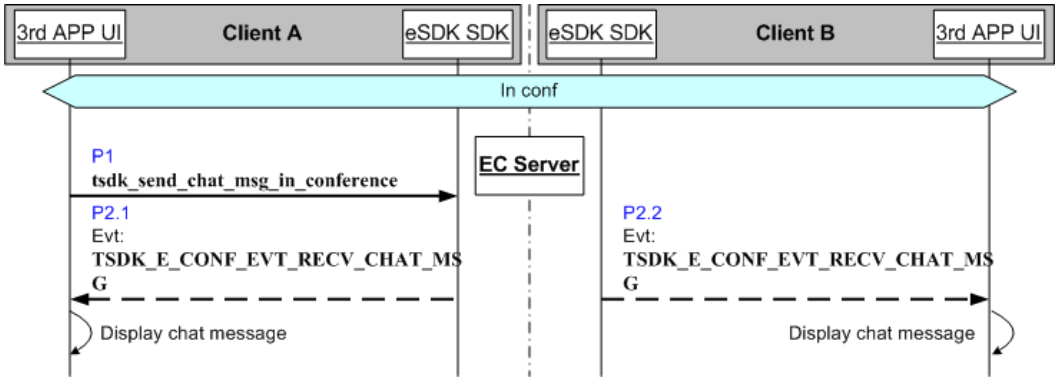
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 19.1.0	CloudVC V600R019C10	
变更	NA	NA	NA

前提条件

1、加入数据会议成功。

流程说明

图 5-28 发送聊天消息



步骤1 UI调用接口tsdk_send_chat_msg_in_conference()接口在会议中发送公共即时消息。

代码示例：

```
TSDK_RESULT ret;
ret = tsdk_send_chat_msg_in_conference(confHandle, chatMsgInfo);
if (TSDK_SUCCESS != ret)
{
    LOG_D_CALL_ERROR("send chat msg failed. result=%#x", ret);
    return -1;
}
return TSDK_SUCCESS;
```

步骤2 会议中所有用户(包括消息发送者)侧SDK均收到消息通知，向UI上报TSDK_E_CONF_EVT_RECV_CHAT_MSG事件，UI显示公共即时消息。

代码示例：

```
case TSDK_E_CONF_EVT_RECV_CHAT_MSG:
{
    /*Notify UI*/
}
```

----结束

注意事项

无。

5.5.5.3 观看白板共享

应用场景

会议中，移动与会者观看白板共享。

说明

移动应用程序暂不具备共享白板的能力，白板的共享者为PC应用程序。

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

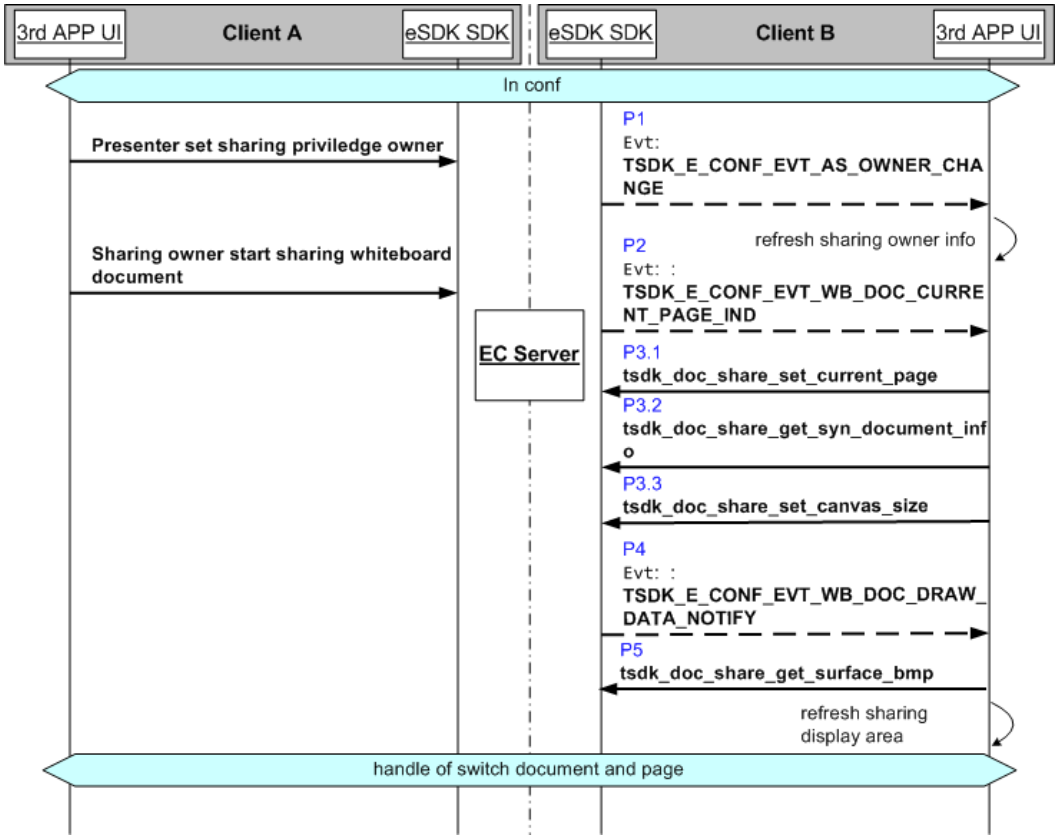
前提条件

1. 加入数据会议成功；
2. 加载白板共享模块成功。

流程说明

开始观看白板共享

图 5-29 开始观看白板共享流程



步骤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白板，移动与会者侧SDK向UI上报共享者变更消息，对应的消息ID为TSDK_E_CONF_EVT_AS_OWNER_CHANGE，携带当前共享权限拥有者ID，UI刷新共享者信息

步骤2 共享者开始共享白板，移动与会者侧SDK上报同步翻页预先通知消息，对应的消息ID为TSDK_E_CONF_EVT_WB_DOC_CURRENT_PAGE_IND

代码示例：

```
case TSDK_E_CONF_EVT_WB_DOC_CURRENT_PAGE_IND:
{
    TSDK_S_DOC_PAGE_BASE_INFO *pageInfo = (TSDK_S_DOC_PAGE_BASE_INFO *)notify.data;
    [self handleDsDocCurrentPageInfoWithConfHandle:notify.param1 andPageInfo:pageInfo];
}
    break;
```

步骤3 UI分别按顺序调用接口tsdk_doc_share_set_current_page()，tsdk_doc_share_get_syn_document_info()，tsdk_doc_share_set_canvas_size()，设置显示区域大小

代码示例：

```
- (void)handleDsDocCurrentPageInfoWithConfHandle:(TSDK_INT32)confHandle andPageInfo:
(TSDK_S_DOC_PAGE_BASE_INFO *)pageInfo
{
    tsdk_doc_share_set_current_page(confHandle, pageInfo, NO);
    TSDK_S_DOC_PAGE_DETAIL_INFO detailInfo;
    memset(&detailInfo, 0, sizeof(TSDK_S_DOC_PAGE_DETAIL_INFO));
    TSDK_RESULT result = tsdk_doc_share_get_syn_document_info(confHandle, pageInfo->component_id,
&detailInfo);
    if (result == TSDK_SUCCESS && (detailInfo.height > 0 && detailInfo.width > 0))
    {
        TSDK_S_SIZE size;
        memset(&size, 0, sizeof(TSDK_S_DOC_PAGE_BASE_INFO));
        size.width = detailInfo.width;
        size.high = detailInfo.height;
        tsdk_doc_share_set_canvas_size(confHandle, pageInfo->component_id, &size, YES);
}}
```

步骤4 共享侧SDK自动抓取白板数据，由业务服务器发送给其他与会者，移动与会者侧SDK向UI上报白板界面数据更新通知消息，对应的消息ID为TSDK_E_CONF_EVT_WB_DOC_DRAW_DATA_NOTIFY。

代码示例：

```
case TSDK_E_CONF_EVT_WB_DOC_DRAW_DATA_NOTIFY:
{
    [self handleWbDocShareDataConfHandle:notify:notify.param1];
}
    break;
```

步骤5 UI调用tsdk_doc_share_get_surface_bmp()接口获取白板数据，刷新共享显示区域。

说明

白板共享数据通过接口同步返回，转换成图片格式在UI显示。

代码示例：

```
- (void)handleWbDocShareDataConfHandle:(TSDK_UINT32)confHandle
{
    __weak typeof(self) weakSelf = self;
    dispatch_async(espace_dataconf_datashare_queue, ^{
        @autoreleasepool {
            TSDK_UINT32 iWidth = 0;
            TSDK_UINT32 iHeight = 0;
            TSDK_VOID *pData = tsdk_doc_share_get_surface_bmp(confHandle, TSDK_E_COMPONENT_DS,
&iWidth, &iHeight);
            if (NULL == pData) {
            DDLogInfo(@"[Meeting] Data is null.");
            return;
            }
            char *pBmpData = (char *)pData;
            TSDK_UINT32 wSize = *(TSDK_UINT32 *)((char *)pBmpData + sizeof(TSDK_UINT16));
            NSData *imgData = [NSData dataWithBytes:(void*)pBmpData length:wSize];
            if (nil == imgData) {
            DDLogInfo(@"[Meeting] Make image from data failed.");
```

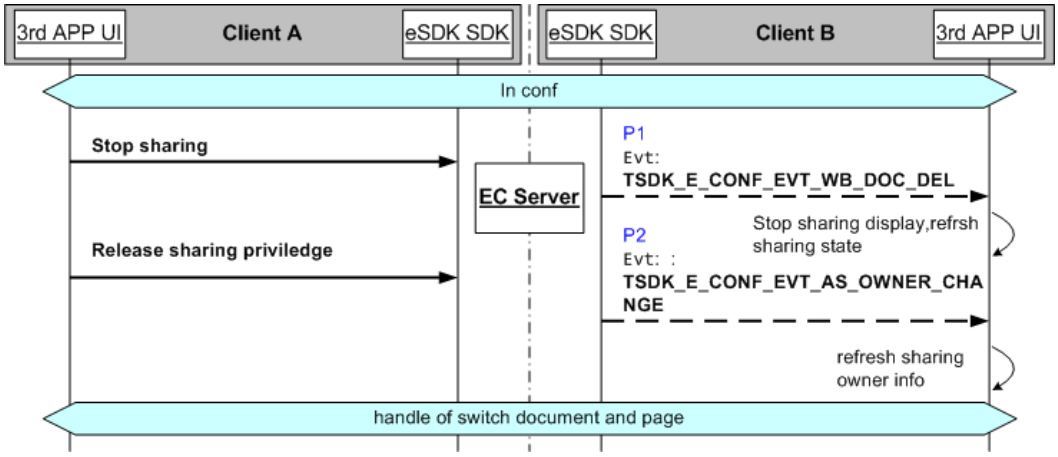


```
            return;
            }
            [weakSelf receiveSharedData:imgData];
        }
    });
}
```

----结束

白板共享结束处理

图 5-30 白板共享结束处理流程



说明

当前图示为共享者主动停止共享，若主讲人结束共享者共享，则对于观看侧，相应的消息顺序相反。

步骤1 共享者停止共享，移动与会者侧SDK通过上报白板共享结束通知消息，对应的消息ID为TSDK_E_CONF_EVT_WB_DOC_DEL，UI刷新界面结束共享。

步骤2 共享者释放共享权限，移动与会者侧SDK向UI上报共享者变更消息，对应的消息ID为TSDK_E_CONF_EVT_AS_OWNER_CHANGE，UI刷新共享者信息。

----结束

注意事项

无。

5.5.5.4 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辅流的能力，辅流的共享者为PC应用程序或硬终端。

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

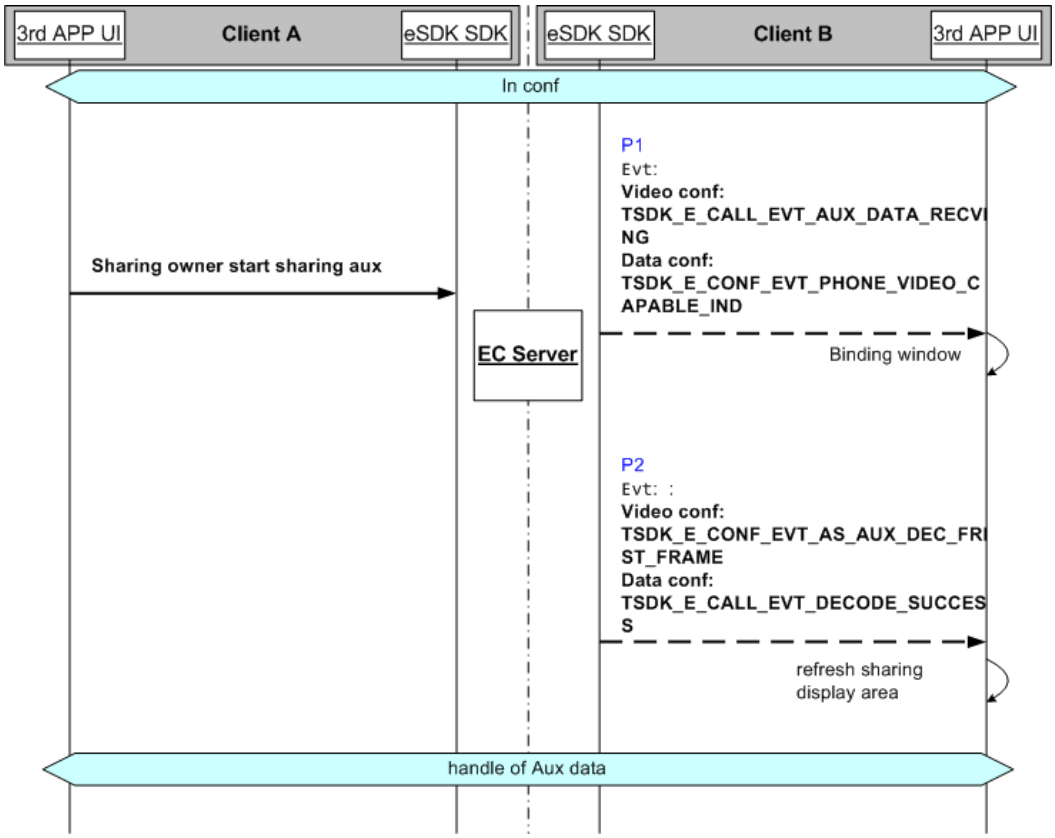
前提条件

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 5-31 开始观看辅流共享流程



步骤1 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视频会议：TSDK_E_CALL_EVT_AUX_DATA_RECVING 数据会议：TSDK_E_CONF_EVT_PHONE_VIDEO_CAPABLE_IND

步骤2 UI调用接口数据会议tsdk_app_share_attach绑定窗口 视频会议在绑定视频窗口的同时绑定辅流窗口

步骤3 共享侧SDK自动抓取辅流数据，由业务服务器发送给其他与会者，移动与会者侧SDK向UI上报辅流解码成功的通知：数据会议
TSDK_E_CONF_EVT_AS_AUX_DEC_FRIST_FRAME 视频会议
TSDK_E_CALL_EVT_DECODE_SUCCESS 展示绑定的辅流界面

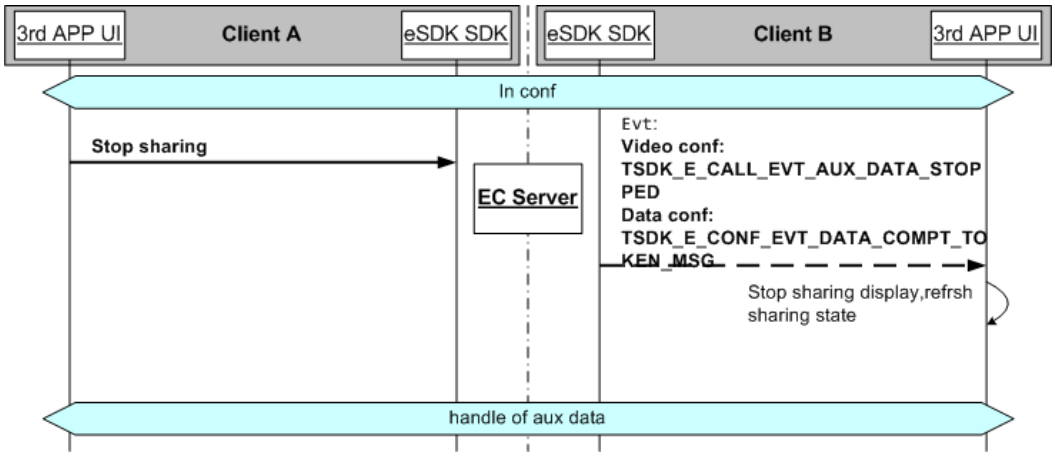
说明

视频会议接收到通知TSDK_E_CALL_EVT_DECODE_SUCCESS 解码成功之后，刷新UI界面；
数据会议接收到通知TSDK_E_CONF_EVT_AS_AUX_DEC_FRIST_FRAME 刷新UI界面。

----结束

辅流共享结束处理

图 5-32 辅流共享结束处理流程



说明

视频会议接收到通知TSDK_E_CALL_EVT_AUX_DATA_STOPPED是停止共享操作，刷新UI界面；
数据会议接收到通知TSDK_E_CONF_EVT_DATA_COMPT_TOKEN_MSG类型结构体中数据TSDK_S_CONF_TOKEN_MSG令牌消息类型中令牌释放指示消息来调用tsdk_app_share_detach(屏幕共享中，观看退出某个数据通道)。

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：TSDK_E_CALL_EVT_AUX_DATA_STOPPED 数据会议：TSDK_E_CONF_EVT_DATA_COMPT_TOKEN_MSG，UI刷新界面结束共享。

----结束

注意事项

视频会议共享时，存在切到后台的操作，使用辅流控制接口tsdk_aux_data_control暂停辅流接收，切回继续接收。

5.6 查询企业通讯录

应用场景

用户查询企业通讯录。

eSDK CloudVC 20.1.RC1 开发指南(iOS)

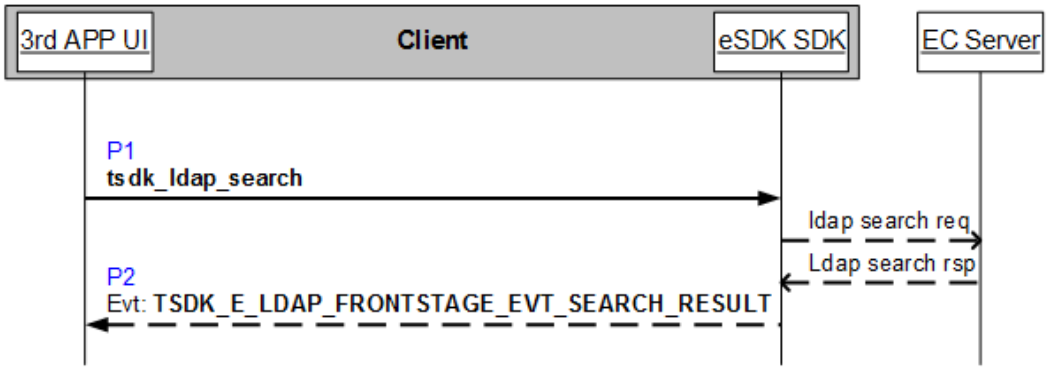
变更记录	eSDK版本	解决方案版本	变更内容
首次发布	eSDK VC 20.1.0	CloudVC V600R020C10	
变更	NA	NA	NA

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 5-33 查询企业通讯录流程



步骤1 UI调用tsdk_ldap_search()发起查询，参数结构为TSDK_S_SEARCH_CONDITION，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TSDK_S_SEARCH_CONDITION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. page_size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. curent_base_dn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TSDK_S_LDAP_SEARCH_RESULT_INFO中的corpName_字段获取。
   4. cookie_length和page_cookie参数在分页查询时，第一次查询时cookie_length为0、page_cookie为空字符串，在后续分页查询时为上次返回的查询结果TSDK_S_LDAP_SEARCH_RESULT_INFO结构中的cookie_len和page_cookie。
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TSDK_E_LDAP_FRONTSTAGE_EVT_SEARCH_RESULT，对应的结果数据结构为TSDK_S_LDAP_SEARCH_RESULT_INFO。

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TSDK_S_LDAP_SEARCH_RESULT_INFO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TSDK_S_SEARCH_CONDITION结构体中的keywords为空字符串，返回的TSDK_S_LDAP_SEARCH_RESULT_INFO结构中，可能会包含curent_base_dn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TSDK_S_LDAP_CONTACT结构体，当TSDK_S_LDAP_CONTACT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。TSDK_S_LDAP_CONTACT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。

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	TSDK_E_MANAGER_ERR_ID	业务管理模块错误码
登录与注销	TSDK_E_LOGIN_ERR_ID	LOGIN 模块错误码
音视频呼叫	TSDK_E_CALL_ERR_ID	CALL模块通用错误码
IPT业务(点击呼叫)	TSDK_E_CTD_ERR_ID	CTD模块错误码
会议	TSDK_E_CONF_ERR_ID	音视频会议模块错误码
企业通讯录	TSDK_E_EADDR_ERR_ID	EADDR 模块错误码
即时消息	TSDK_E_IM_ERR_ID	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"tsdk_"，"tup_"或"hme_"为前缀。

日志分析

1. SDK日志的一般格式为：

   [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
```


eSDK CloudVC 20.1.RC1 开发指南(iOS)

7 附录

7.1 HelloWorld 文件源码

7.1.1 ViewController.mm

```
#import "ViewController.h"

#import "tsdk_login_def.h"
#import "tsdk_login_interface.h"
#import "tsdk_error_def.h"
#import "tsdk_manager_interface.h"
#import "tsdk_manager_def.h"
#include <netdb.h>
#include <net/if.h>
#include <ifaddrs.h>
#include <arpa/inet.h>
#include <dlfcn.h>
#include <sys/sysctl.h>
@interface ViewController ()
@property (strong, nonatomic)  UITextField *account_textField;
@property (strong, nonatomic)  UITextField *password_textField;
@property (strong, nonatomic)  UITextField *serverAddress_textField;
@property (strong, nonatomic)  UITextField *serverport_textField;
@property (strong, nonatomic)  UIButton *login_button;
@end
@implementation ViewController
-(void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    [self configView];
}
- (void)viewDidLoad
{
    [super viewDidLoad];
    [self initUportalLoginService];
}
-(void)configView
{
    _account_textField = [[UITextField alloc] init];
    _account_textField.borderStyle = UITextBorderStyleRoundedRect;
    _account_textField.center = CGPointMake(self.view.bounds.size.width/2, 60);
```

eSDK CloudVC 20.1.RC1 开发指南(iOS)


```
    _account_textField.bounds = CGRectMake(0, 0, self.view.bounds.size.width - 20, 30);
    _account_textField.placeholder = @"Account";
    [self.view addSubview:_account_textField];

    _password_textField = [[UITextField alloc] init];
    _password_textField.borderStyle = UITextBorderStyleRoundedRect;
    _password_textField.center = CGPointMake(self.view.bounds.size.width/2,
CGRectGetMaxY(_account_textField.frame)+30);
    _password_textField.bounds = CGRectMake(0, 0, self.view.bounds.size.width - 20, 30);
    _password_textField.placeholder = @"Password";
    [self.view addSubview:_password_textField];

    _serverAddress_textField = [[UITextField alloc] init];
    _serverAddress_textField.borderStyle = UITextBorderStyleRoundedRect;
    _serverAddress_textField.center = CGPointMake(self.view.bounds.size.width/2,
CGRectGetMaxY(_password_textField.frame)+ 30);
    _serverAddress_textField.bounds = CGRectMake(0, 0, self.view.bounds.size.width - 20, 30);
    _serverAddress_textField.placeholder = @"Server Address";
    [self.view addSubview:_serverAddress_textField];

    _serverport_textField = [[UITextField alloc] init];
    _serverport_textField.borderStyle = UITextBorderStyleRoundedRect;
    _serverport_textField.center = CGPointMake(self.view.bounds.size.width/2,
CGRectGetMaxY(_serverAddress_textField.frame)+30);
    _serverport_textField.bounds = CGRectMake(0, 0, self.view.bounds.size.width - 20, 30);
    _serverport_textField.placeholder = @"Server Port";
    [self.view addSubview:_serverport_textField];

    _login_button = [[UIButton alloc] init];
    _login_button.center = CGPointMake(self.view.bounds.size.width/2,
CGRectGetMaxY(_serverport_textField.frame)+30);
    _login_button.bounds = CGRectMake(0, 0, self.view.bounds.size.width - 20, 30);
    [_login_button setBackgroundColor:[UIColor redColor]];
    [_login_button setTitle:@"Login" forState:UIControlStateNormal];
    [_login_button setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];
    [_login_button addTarget:self action:@selector(loginAction)
forControlEvents:UIControlEventTouchUpInside];
    [self.view addSubview:_login_button];
}
- (void)loginAction
{
    [self.view endEditing:YES];
    BOOL result = [self loginAuthorizeWithServerAddress:_serverAddress_textField.text port:
[_serverport_textField.text intValue] account:_account_textField.text password:_password_textField.text];
    if (!result)
    {
        [self showMessage:@"Uportal login fail!"];
    }
}
#pragma mark - Service
TSDK_VOID onTSDKNotifications(TSDK_UINT32 msgid, TSDK_UINT32 param1, TSDK_UINT32 param2,
TSDK_VOID *data)
{
    NSLog(@"onTUPLoginNotifications : %#x",msgid);
    dispatch_async(dispatch_get_main_queue(), ^{
        switch (msgid)
        {
            case TSDK_E_LOGIN_EVT_AUTH_SUCCESS:
            {
                NSLog(@"Uportal login success !");
                [ViewController showMessages:@"Uportal login success"];
            }
                break;
            case TSDK_E_LOGIN_EVT_AUTH_FAILED:
            {
                NSLog(@"Uportal login fail !");
                [ViewController showMessages:@"Uportal login fail"];
            }
                break;
```



```
            default:
                break;
        }
    });
}

-(BOOL)initUportalLoginService
{
    TSDK_S_LOG_PARAM logParam;
    memset(&logParam, 0, sizeof(TSDK_S_LOG_PARAM));
    NSString *logPath = [[NSHomeDirectory() stringByAppendingPathComponent:@"Documents"]
stringByAppendingString:@"/TUPC60log"];
    NSString *path = [logPath stringByAppendingString:@"/tsdk"];
    logParam.level = TSDK_E_LOG_DEBUG;
    logParam.file_count = 1;
    logParam.max_size_kb = 4*1024;
    strcpy(logParam.path, [path UTF8String]);
    TSDK_RESULT result = tsdk_set_config_param(TSDK_E_CONFIG_LOG_PARAM, &logParam);

    TSDK_S_APP_INFO_PARAM app_info;
    memset(&app_info, 0, sizeof(TSDK_S_APP_INFO_PARAM));
    app_info.client_type = TSDK_E_CLIENT_MOBILE;
    strcpy(app_info.product_name, "SoftClient on Mobile");
    app_info.support_audio_and_video_call = TSDK_TRUE;
    app_info.support_ctd = TSDK_TRUE;
    app_info.support_audio_and_video_conf = TSDK_TRUE;
    app_info.support_enterprise_address_book = TSDK_TRUE;
    result = tsdk_init(&app_info ,&onTSDKNotifications);
    return result == TSDK_SUCCESS ? YES : NO;

}
-(BOOL)loginAuthorizeWithServerAddress:(NSString *)serverAddress port:(int)port account:(NSString
*)account password:(NSString *)password
{
    TSDK_S_LOGIN_PARAM loginParam;
    memset(&loginParam, 0, sizeof(TSDK_S_LOGIN_PARAM));
    loginParam.user_id = 1;
    loginParam.auth_type = TSDK_E_AUTH_NORMAL;
    strcpy(loginParam.user_name, [account UTF8String]);
    strcpy(loginParam.password, [password UTF8String]);
    loginParam.server_type = TSDK_E_SERVER_TYPE_PORTAL;
    strcpy(loginParam.server_addr, [serverAddress UTF8String]);
    loginParam.server_port = (TSDK_UINT16)port;
    TSDK_RESULT result = tsdk_login(&loginParam);

    return result == TSDK_SUCCESS ? YES : NO;
}
-(void)showMessage:(NSString *)msg
{
    [[[UIAlertView alloc] initWithTitle:nil message:msg delegate:nil cancelButtonTitle:nil
otherButtonTitles:@"OK", nil] show];
}
+(void)showMessages:(NSString *)msg
{
    [[[UIAlertView alloc] initWithTitle:nil message:msg delegate:nil cancelButtonTitle:nil
otherButtonTitles:@"OK", nil] show];
}
-(void)touchesBegan:(NSSet<UITouch *> *)touches withEvent:(UIEvent *)event
{
    [self.view endEditing:YES];
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
-(void)dealloc
{
    [[NSNotificationCenter defaultCenter] removeObserver:self];
```

```
}
@end
```

7.2 全平台操作系统要求

eSDK CloudEC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(iOS)

8 修订记录

发布日期	文档版本	修订说明
2019-06-14	06	配套19.1.1版本文档刷新发布。
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、新增对融合会议On-premise组网支持； 2、新增会议中发送消息功能；
2018-05-10	01	配套6.1.0.RC1版本(暂仅支持融合会议Hosted组网)，文档首次发布。