

eSDK eLTE V200R001C00

开发指南 01(PC,API,C++)

文档版本 01

发布日期 2016-10-25



版权所有 © 华为技术有限公司 2016。 保留一切权利。

非经本公司书面许可,任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部,并不得以任何形式传播。

商标声明



HUAWEI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: http://www.huawei.com
客户服务邮箱: support@huawei.com

客户服务电话: 4008302118

目 录

1 eSDK eLTE SDK 是什么	1
2 内容导航	3
3 相关资源	4
3.1 eSDK eLTE 社区首页	5
3.2 SDK 下载路径	5
3.3 Sample Codes	5
3.4 接口参考	6
3.5 IDE 插件	6
3.6 免费申请远程实验室	7
3.7 SDK 修订记录	9
3.8 技术支持渠道	9
4 Hello World	10
4.1 资源准备	11
4.2 安装本地服务	11
4.3 创建工程	13
4.4 编写代码	18
4.5 编译调试	20
5 初始化配置	23
5.1 设置日志路径	24
5.2 获取 SDK 版本号	24
5.3 初始化资源	24
5.4 设置回调函数	25
5.5 用户登录 eAPP 系统	25
5.6 触发状态上报	25
6 典型业务开发场景	26
6.1 资源管理	27
6.1.1 概述	27
6.1.2 用户和群组管理	27
6.1.3 获取录音录像文件列表	42
6.2 语音调度	45
6.2.1 概述	45

开发指南 01(PC.API.C++)

71 X 1 1 1 0 1 (1 C, 1 1 1, C · ·)	1 3
6.2.2 语音点呼场景	46
6.2.3 语音组呼场景	55
6.3 视频调度	61
6.3.1 概述	61
6.3.2 视频回传场景	61
6.3.3 视频分发场景	66
6.3.4 视频上墙场景	
6.4 短数据	73
6.4.1 概述	73
6.4.2 发送短数据场景	73
6.5 订阅终端 GIS	77
6.5.1 概述	77
6.5.2 订阅终端 GIS 位置信息	77
7 定位指南	81
8 开发指南修订记录	84

1 eSDK eLTE SDK 是什么

关于 eLTE 宽带集群

华为eLTE宽带集群解决方案,基于最先进的LTE无线宽带技术,上行吞吐量可达 100Mpbs,下行吞吐量达50Mbps。一网支持多媒体集群语音、视频调度,高清无线视 频监控,超远程数据采集和移动办公业务。

该方案是业界首个宽带集群解决方案,同时支持语音和视频调度。其专业的语音性集群组呼时延<300ms,群组抢占时延<150ms,关键时刻值得信赖。同时支持的视频上传,以及分发。调度过程不仅能听到,而且能看到,做到"现场可见,指挥可达",将极大地提升指挥调度效率和快速响应能力,适用于公共安全,交通,电力,能源等多种行业。

详细请见eLTE宽带集群首页。

eSDK eLTE SDK 是什么

华为eSDK eLTE解决方案是对eLTE宽带集群产品eAPP SDK接口进行标准化封装,为合作伙伴提供集群用户管理、群组管理、语音点呼、语音组呼、人工转接、缜密监听、环境监听、实时视频回传与分发、视频上墙、短彩信发送与接收以及集群终端GIS订阅与地理位置信息上报等管理调度功能。

强大的功能和标准化的接口封装使得合作伙伴易于将华为eSDK eLTE SDK与行业的上层应用融合,为政府、交通、教育以及高端企业客户构建安全、便捷以及实用的语音和视频调度平台。

为避免第三方集成多个产品接口时出现底层库冲突问题,eSDK eLTE SDK是将产品的 SDK封装成本地服务和本地API,服务不对外开放,第三方可直接调用本地API。eSDK eLTE分为SDK和OCX两种形式。

如果您使用C#或JavaScript语言进行二次开发,建议使用eSDK eLTE OCX 开发指南(Windows, C++)。当然C#也可以调用SDK动态库,但是涉及到大量的数据转换。

如果您使用C++语言开发,建议直接使用SDK, Windows SDK动态库是基于各种功能 API的封装,详情请参考本开发指南。

eSDK eLTE SDK 包含什么

eSDK eLTE SDK支持在Windows系统上进行二次集成开发,目前支持Window7 32位/64位, Windows8 32位/64位开发环境。主要包含以下内容:

● API SDK包

API本地开发包用于eLTE宽带集群业务二次开发调用。详细请参见SDK下载路径。

Sample Codes

华为SDK提供一系列Sample codes用于指导如何调用SDK去构建应用工程。更多信息请见Sample Codes。

• IDE support

为了给开发者更好的体验,我们提供基于Visual Studio和eclipse的eSDK IDE插件,包含下载并安装Demo、连接远程实验室、在线支持等功能,完成工程创建、配置、调试等任务,简化二次开发过程。更多信息请参见IDE插件。

● Github源码

华为eSDK eLTE二次开发提供基于多语言多形态的源代码。更多代码资源请前往 Github获取。

2 内容导航

开发指南可以帮助您了解如何安装、配置、使用eSDK eLTE SDK调用eAPP系统的业务功能。我们强烈建议您按照以下顺序阅读此指南:

- 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 code下载链接,介绍eLTE IDE插件并演示如何使用,如何申请远程实验室等。
- 2. Hello World:如果你仅仅是要把SDK运行起来,你应该首先看这部分,它会指导你如何下载安装SDK,以及如何配置你的开发环境,并帮助您快速调用初始化以及登录接口,实现登录调度机系统。
- 3. **初始化配置**:业务开发前,需要设置日志路径、获取SDK版本号、初始化资源、设置消息回调函数、登录eAPP服务器、触发状态上报等初始化配置操作。
- 4. **典型业务开发场景**:包括eSDK eLTE SDK开放的典型功能场景,详细介绍开发流程、样例代码以及注意事项等信息。
- 5. 定位指南:帮助您定位开发过程中遇到的常见问题。
- 6. 开发指南修订记录:帮助您了解各版本开发指南更新细节。

阅读建议

- 如果您刚开始了解和使用eSDK eLTE SDK,请您按照开发指南文章顺序阅读。
- 如果您想深入核心业务的二次开发,建议您直接阅读典型开发场景。
- 如果本指南中没有您需要的接口功能,请参考eSDK eLTE API接口参考 (Windows,C++)。
- 如果您在使用eSDK eLTE SDK过程中遇到问题可以参考定位指南自助定位,或者前往华为开发者社区eLTE版块在线FAQ查询,如果仍不能解决您遇到的问题,可以在DevCenter提问题或提需求给我们,我们会尽快答复您。同时也欢迎您联系eSDK eLTE技术支持。

∭ 说明

本开发指南配套的API版本号V2.1.00,如果需要查看其它版本API对应的开发指南,请进入 华为开发者社区资源中心,点击"开发文档>开发指南>宽带集群 > eAPP610 > eSDK eLTE API开发指南",选择对应版本进行下载。

3 相关资源

- 3.1 eSDK eLTE社区首页
- 3.2 SDK下载路径
- 3.3 Sample Codes
- 3.4 接口参考
- 3.5 IDE插件
- 3.6 免费申请远程实验室
- 3.7 SDK修订记录
- 3.8 技术支持渠道

3.1 eSDK eLTE 社区首页

您可通过**华为开发者社区eLTE板块**,快速体验eLTE业务功能,获得eSDK eLTE的二次 开发快读体验、Hello World、开发指南、接口参考、SDK资源下载、FAQ以及技术支持信息。

3.2 SDK 下载路径

进入**华为开发者社区资源中心**,点击 "SDK > **宽带集群** > eAPP610 > eSDK eLTE API **业务包**",选择对应版本进行下载。

3.3 Sample Codes

代码样例目录包含了《典型开发场景》中介绍的5个功能Sample,演示如何使用SDK调用eLTE业务功能。

∭说明

请使用Visual Studio2010及以上版本编译执行Sample Codes。

Sample Codes 列表

语言: C++

Sample Codes名称	功能描述
eLTE_ResourceManage	基于eLTE SDK的C++ Sample,主要展示 华为eSDK eLTE宽带集群系统提供的资 源管理能力,包括用户管理和群组管 理,录音录像文件查询功能。
eLTE_Audio	基于eLTE SDK的C++ Sample,主要展示 华为eSDK eLTE宽带集群系统提供的语 音调度能力,包括语音点呼、语音组 呼、紧急组呼、人工转接、缜密监听、 环境监听等功能。
eLTE_SDK_Video_Demo	基于eLTE SDK的C++ Sample, 主要展示 华为eSDK eLTE宽带集群系统提供的视 频调度能力,包括发起实时视频回传、 视频分发、视频上墙等功能。
eLTE_SDS	基于eLTE SDK的C++ Sample, 主要展示 华为eSDK eLTE宽带集群系统提供的短 数据能力,包括对用户和群组发送短信 和彩信,接收短信和彩信功能,
eLTE_SDK_GIS_Demo	基于eLTE SDK的C++ Sample,主要展示 华为eSDK eLTE宽带集群系统提供的订 阅终端GIS能力,可以通过订阅终端GIS 获得终端周期上报的地理位置信息。

∭说明

您也可以进入**华为开发者社区资源中心**,点击"**示例>宽带集群 >eAPP610>eSDK eLTE API 代码样例",选择对应版本进行下载。**

3.4 接口参考

接口参考主要包含:

概述:明确配套版本、使用背景、场景、前提条件等内容;并指出可以获取哪些信息,完成什么样的功能。

接口详细描述:

- 接口描述:详细描述该接口功能、应用场景和使用方法。
- 使用说明:描述该接口函数使用时需要注意的事项、使用的限制;功能相似的接口或者需要配合使用的接口;前提条件等。
- 方法定义:接口函数的完整声明。
- 参数描述:详细描述参数的含义、取值范围、使用限制、参数之间的依赖关系等。
- 返回值:说明该接口函数的返回值。
- 使用示例:举例说明该接口函数的使用,关键代码需要注释。

错误码

完整的错误码列表,包含开发过程中所有可能出现的错误码及描述。

□ 说明

您可以进入<mark>华为开发者社区资源中心</mark>,点击"**开发文档 > 接口文档 > 宽带集群 > eAPP610 > eSDK eLTE API接口参考**",选择对应版本进行下载。

3.5 IDE 插件

我们提供配套Visual Studio的IDE工具插件,支持如下特色功能:

- 一键式下载及安装SDK
 - 一键式下载安装eSDK各业务模块SDK包,SDK管理更简单。
- 随时随地接入远程实验室 随时随地接入远程实验室进行在线联调,实验室接入更方便。
- 创建向导Demo

Demo创建向导,快速完成Demo配置并接入远程实验室调测,体验更快捷。

- 新建工程更简单
 - 快速完成新建工程,且提供包管理器下载好的包引用操作,无需再次手动添加。
- 在线支持"0"等待 "0"等待支持,即时消息,语音支持,沟通更高效。
- 在线文档一键查询 SDK文档一键查询,资料查阅更省心。

利用eSDK IDE插件,能够快速配置eLTE Demo工程,以及自动安装eLTE SDK包,二次开发效率更高。如果您想快速了解IDE工具,请参考IDE快速入门。

3.6 免费申请远程实验室

华为 eSDK 远程实验室简介

华为远程实验室为开发者提供了7×24小时的免费的云化实验室环境,提供真实的华为设备供开发者远程在线开发调试。借助远程实验室自助管理平台,开发者不需要购置华为产品便可基于远程实验室针对相关产品进行二次开发,并实现远程对接测试认证。目前华为远程实验室已发布云计算、SDN、大数据、统一通信与协作、BYOD、eLTE、敏捷网络等7个生态圈的26个实验室环境。

相关信息请查询华为开发者社区远程实验室。

远程实验室有哪些优势

- 低门槛: 官网注册用户即可申请使用,环境与预约时长、预约次数受限;
- 分级支持:环境按域划分,重点开发者、合作伙伴可访问特定环境并享受额外的 预约环境:
- 全球资源高速互联:建设以苏州远程实验室为中心的全球实验室分布格局,依托 IT全球100ms高性能骨干网络和IT全球端到端应用保障能力。

如何免费申请远程实验室

申请步骤如下:



步骤1 注册账号。

- 1. 若您已有华为注册账号,则直接在登录远程实验室首页使用已注册账号登录。
- 2. 若您还未注册华为账号,请进入**华为远程实验室网站**,点击"进入实验室",页面跳转至账号注册页,填写注册信息,并在注册邮箱中激活您的华为账号。

步骤2 预约环境。

- 1. 若您已成功预约环境,且环境可用时长在有效期范围内,则请忽略此步骤。
- 2. 华为账号注册成功后,系统会自动跳转到华为远程实验室首页,状态为已登录。 找到eLTE环境,点击"预约"按钮,默认预约的环境可用时长为2小时,如下图所示。



- 3. 预约成功后,系统会自动发送VPN接入地址、用户名、密码等信息至您的华为账户注册邮箱中,根据邮件中的提示信息下载HUAWEI VPN客户端软件,完成本地安装。
- 4. 在我的预约中进入eLTE环境页面,查看user-3属性,获取Login_Name和Login_Password。

步骤3 接入环境。

1. 打开SVN客户端,将预约成功后系统发送给您的VPN地址、用户名、密码等信息,分别输入到客户端对应的输入框内,并点击"登录"。



2. 连接成功后会在系统状态栏提示"VPN连接成功"。

步骤4 调测发布。

在使用申请成功的eLTE调度员账号、密码、IP地址、Port等信息,完成登录,调测您的应用程序。

----结束

3.7 SDK 修订记录

SDK会逐渐升级来支持新的服务。您可以参考开发者社区网站历史版本链接了解不同版本有哪些更新。具体的更新信息包含:

- 版本信息:提交者、最新版本、创建时间、最近更新。
- 下载链接;
- 特性列表;
- 依赖升级;
- 移除特性;
- 支持的API版本,版本发布时间;

3.8 技术支持渠道

- 开发过程中,您有任何问题可以在DevCenter用户提单中心中提单跟踪。
- 如果您在远程实验室使用过程中有任何疑问,都可以通过以下方式联系我们。
 - a. 华为技术支持热线电话: 400-882-8000
 - b. 华为技术支持邮箱: esdk@huawei.com

4 Hello World

本示例以C++语言进行eSDK eLTE API 的二次集成开发,实现调度员用户登录调度系统。

开发过程中的故障处理请参考定位指南。

- 4.1 资源准备
- 4.2 安装本地服务
- 4.3 创建工程
- 4.4 编写代码
- 4.5 编译调试

4.1 资源准备

在Hello World开始之前,您需要准备如下环境与资源。

开发工具

- 操作系统: Windows 7 Professional。
- Microsoft Visual Studio: 建议使用Visual Studio 2010 专业版及以上版本。
- Microsoft Visual Studio Service Pack 1: 在Windows 7专业版 PC上安装Visual Studio 2010 专业版和Visual Studio 2010 Service Pack 1软件,详细安装方法请参考Visual Studio官网。

SDK 资源

SDK业务包名称: eSDK eLTE API V2.1.00.zip

SDK业务包下载路径: 您可以进入**华为开发者社区资源中心**,点击"SDK > **宽带集群** > **eAPP610** > **eSDK** eLTE API业务包",选择对应版本进行下载。

🔲 说明

V2.1.00表示华为eSDK eLTE版本号, 如: V2.1.00表示版本号为V200R001C00。

eLTE 环境

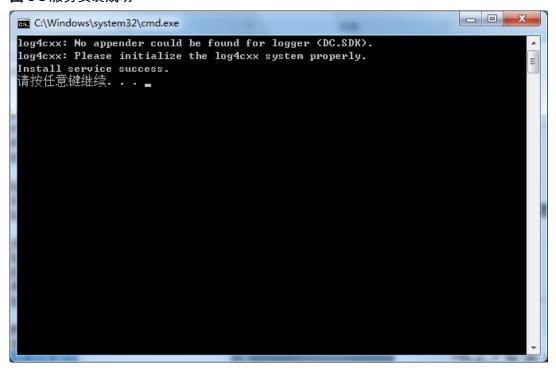
若您还未购买华为eLTE宽带集群产品,可登录华为远程实验室,**免费申请远程实验室** 环境,体验eLTE宽带集群和调试二次开发代码。

4.2 安装本地服务

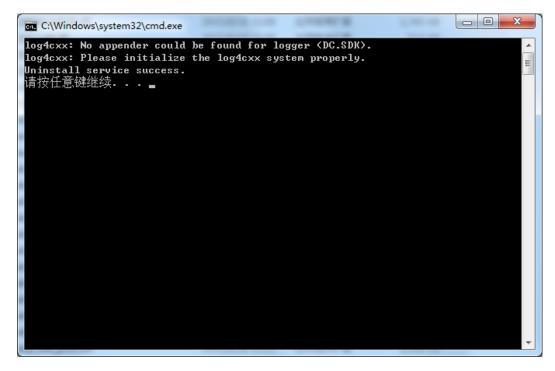
步骤1 将下载的二次开发业务包 "eSDK_eLTE_OCX_V2.1.00.zip"解压缩到项目工程目录中。

步骤2 以管理员身份运行项目工程目录下的"eLTE_NativeService/InstallService.bat"文件,系统弹出如下所示提示信息,表示服务安装成功。

图 4-1 服务安装成功



步骤3 如果想卸载服务,则以管理员的身份运行InstallService.bat同级目录下的 UninstallService.bat脚本即可,系统弹出如下图所示提示信息,则表示服务卸载成功。

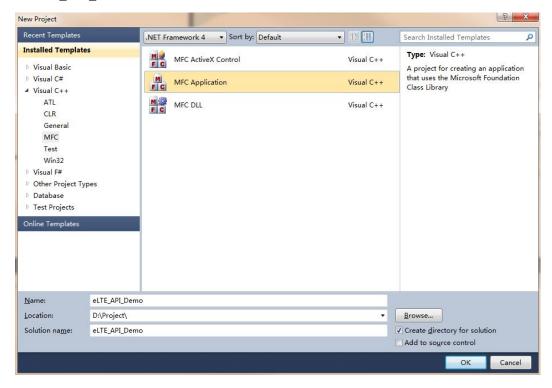


----结束

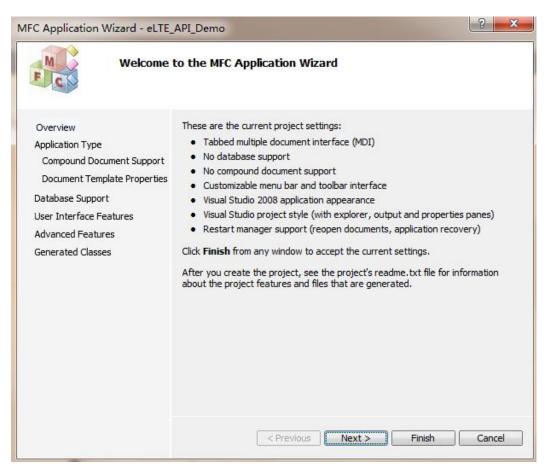
4.3 创建工程

步骤1 打开"Windows Visual Studio 2010",选择"**File** > **New** > **Project**"。系统显示"New Project"界面。

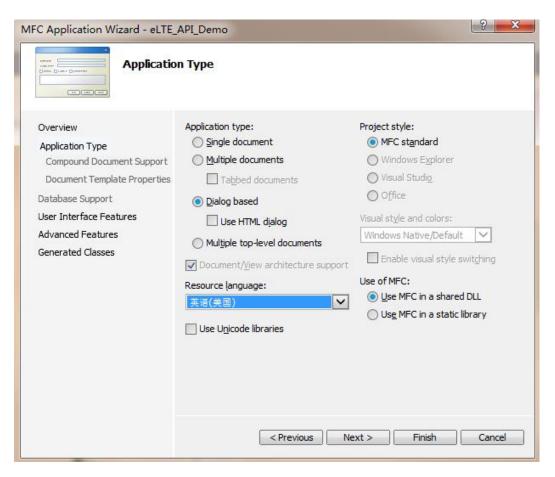
步骤2 选择 "Visual C++>MFC>MFC Application",输入项目名称 "eLTE_API_Demo"。



步骤3 单击"OK",进入MFC应用程序向导,系统显示界面如下。

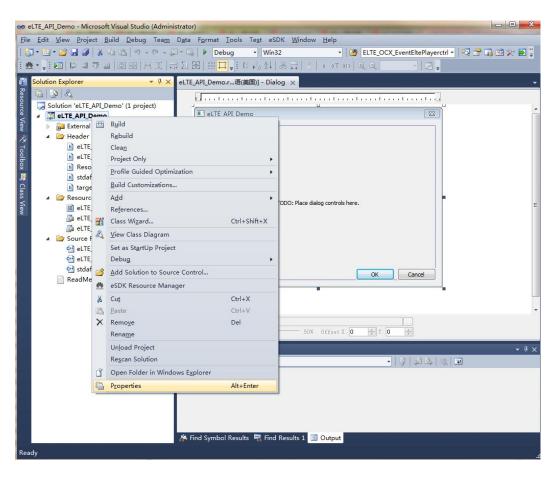


步骤4 单击"Next",进入MFC应用程序类型窗口,选择应用程序类型为"Dialog based",取消勾选"Use Unicode libraries",系统显示界面如下。

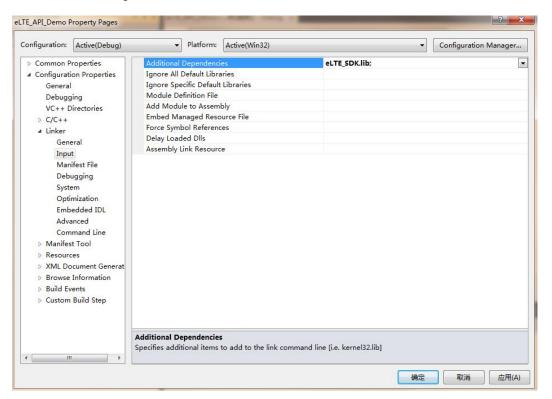


步骤5 单击"Finish",完成新工程的创建。

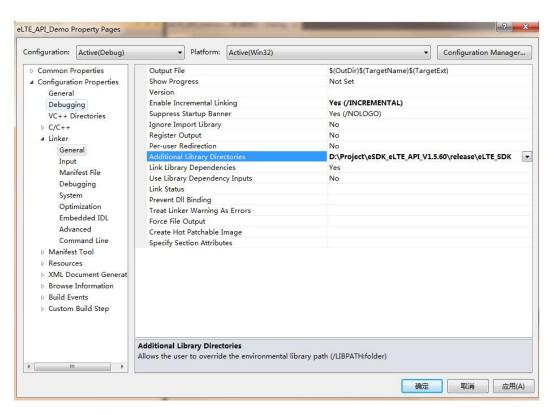
步骤6 右击工程选择"Properties",如下图所示。



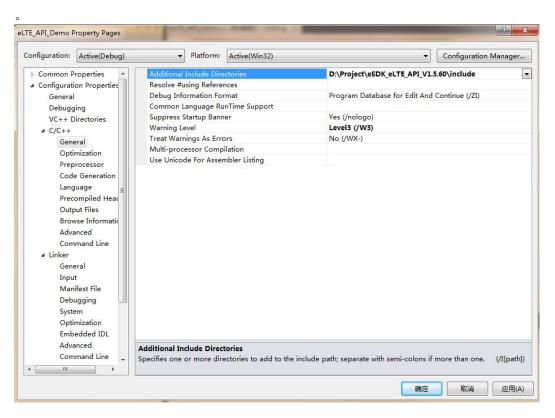
步骤7 选择"Linker > Input",添加库文件。



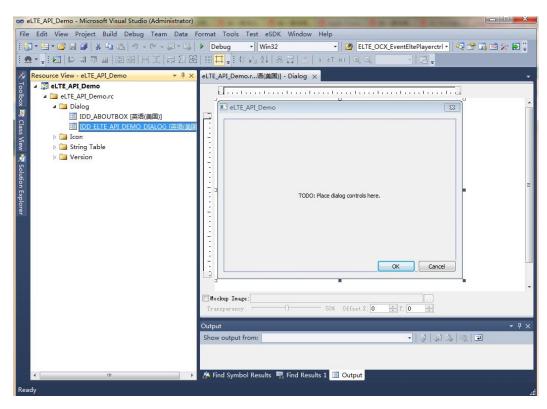
步骤8 选择"Linker > General",添加库文件路径。



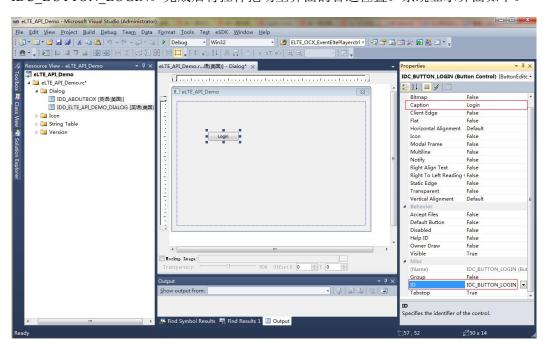
步骤9 选择 "C/C++> General",添加头文件 "eLTE_Types.h"和 "eLTE_SDK.h"路径。



步骤10 打开Resource View,选择"eLTE_API_Demo > eLTE_API_Demo.rc",双击 "IDD_ELTE_API_DEMO_DIALOG",如下图所示。



步骤11 删除IDD_ELTE_API_DEMO_DIALOG中自动生成的控件,打开Toolbox,添加登录Button,右键单击Button,选择**Properties**,修改**Caption**为Login以及将其**ID**改成IDC BUTTON LOGIN。完成后将控件拖动至界面的合适位置。系统显示界面如下。



----结束

4.4 编写代码

步骤1 在stdafx.h 里添加头文件,代码如下。

```
#include "eLTE_SDK.h"
#include "eLTE_Types.h"
#include <string>
```

步骤2 在eLTE_API_DemoDlg.h 的CeLTE_API_DemoDlg类里添加事件回调声明,代码如下。

```
public:
// 事件回调函数
static ELTE_VOID __SDK_CALL ELTE_EventCallBack(ELTE_INT32 iEventType, ELTE_VOID* pEventBuf,
ELTE_UINT32 uiBufSize, ELTE_VOID* pUserData);
```

步骤3 在eLTE API DemoDlg.cpp 里添加处理回调信息,代码如下。

```
ELTE_VOID __SDK_CALL CeLTE_API_DemoDlg::ELTE_EventCallBack(ELTE_INT32 iEventType, ELTE_VOID*
pEventBuf, ELTE_UINT32 uiBufSize, ELTE_VOID* pUserData)
{
//回调信息中找到<StatusValue>4011</StatusValue>字符串则输出Login Success
char* pEventXml = (char*)pEventBuf;
CString xmlStr(pEventXml);
CeLTE_API_DemoDlg* pDlg = (CeLTE_API_DemoDlg*)pUserData;
int i = xmlStr.Find("<StatusValue>4011</StatusValue>");
if(ELTE_Event_NotifyResourceStatus == iEventType&&i!=-1)
{
    pDlg->MessageBox("Login Success");
}
```

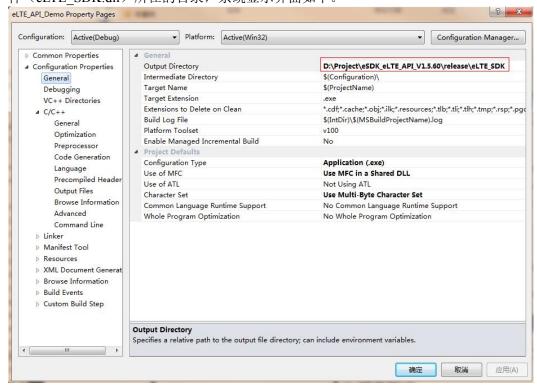
步骤4 双击Login Button,添加代码,参考如下。

```
// 初始化SDK
  ELTE INT32 iRet = ELTE SDK Init(FALSE);
  if(0 != iRet)
    MessageBox(_T("ELTE_SDK_Init failed."));
   // 设置回调函数
  iRet = ELTE_SDK_SetEventCallBack(ELTE_EventCallBack, this);
  if(0 != iRet)
    {\tt MessageBox}(\_{\tt T("ELTE\_SDK\_SetEventCallBack\ failed."));}
  //具体登录信息需要按照实际情况自行配置
   //用户名
  CString strUserName = ("4120");
   //用户密码
  CString strPWD = ("4120");
  //服务器IP
  CString strServerIP = ("172.22.9.120");
  //本地IP地址
  CString strLocalIP = ("172.23.0.32");
  //端口号
  int SipPort = 5060;
  //登录
  iRet = ELTE_SDK_Login(strUserName, strPWD, strServerIP, strLocalIP, SipPort);
  if(0 != iRet)
    MessageBox("ELTE_SDK_Login failed.");
    return;
```

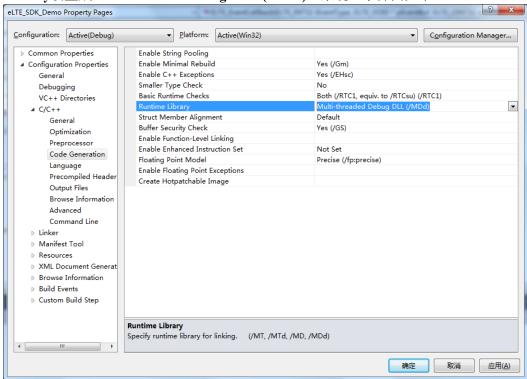
----结束

4.5 编译调试

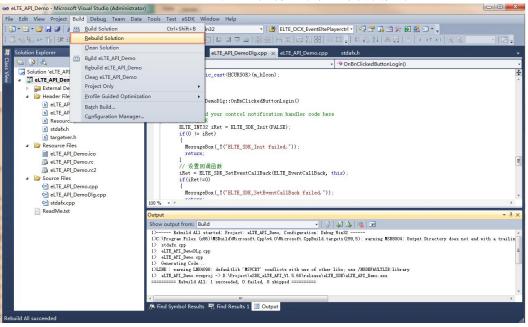
步骤1 右击工程选择 "Properties > Configuration > General",将工程输出路径设置成库文件(eLTE_SDK.dll)所在的目录,系统显示界面如下。



步骤2 右击工程选择 "Properties >C/C++ >Code Generation", 在debug模式下将Runtime Library设置成Multi-threaded Debug DLL (/MDd), 系统显示界面如下。



步骤3 点击"Build > Rebuild Solution"生成.exe文件,系统显示界面如下。



步骤4 若您已购买华为eLTE产品,请直接在eLTE_API_DemoDlg.CPP中填写登录产品的用户 名、密码、服务器IP、本机IP以及服务器端口进行代码调试运行。

若您还未购买华为eLTE产品,可登录**华为远程实验室**,免费申请调测环境,运行您所编辑的代码。

这里调试以用户登录接口为例。

在代码内事先写入正确的用户名、密码、服务器IP、本机IP以及服务器端口。

在对应的输出路径(如D:\Project\eSDK_eLTE_API_V2.1.00\release\eLTE_SDK)中运行eLTE_API_Demo.exe,点击"Login",页面显示"**Login Success**",表示登录成功。如下图所示。



步骤5 若您还未购买华为eLTE产品,可登录华为远程实验室,免费申请调测环境,运行您所编辑的代码。

----结束

5 初始化配置

您需要完成以下初始化配置后再进行eLTE业务二次集成开发,以下配置在后续开发过程中无需再进行,但可修改。

- 5.1 设置日志路径
- 5.2 获取SDK版本号
- 5.3 初始化资源
- 5.4 设置回调函数
- 5.5 用户登录eAPP系统
- 5.6 触发状态上报

5.1 设置日志路径

SDK会默认在运行程序下面生成相应的日志文件,如果您需要指定路径可以调用设置日志路径【ELTE SDK SetLogPath(设置日志路径)】接口。

在开发和测试阶段,我们建议您开启Debug级别日志。Debug级别提供更详细的SDK运行信息和错误提示,更方便定位问题。在版本发布时,我们建议您关闭Debug日志,这样可以减少文件输入输出操作,提高程序运行效率。

∭说明

如果需要自定义日志路径,必须在ELTE_SDK_Init之前调用该接口,否则设置将不生效。不建议使用相对路径,且相对路径不能以\\开头。

代码示例如下:

```
//cpp code
ELTE_INT32 iRet = ELTE_SDK_SetLogPath("D:\\log\\");
if(iRet==0)
{
    //成功
}
```

此外,您还可以通过修改配置文件来设置日志路径和级别,修改二次开发包目录下eSDKClientLogCfg.ini文件,将LogPath=./logs修改为LogPath=指定路径即可,路径尽量使用绝对路径。

5.2 获取 SDK 版本号

调用获取版本号【ELTE_SDK_GetVersion(获取SDK版本号)】接口,获取SDK版本号。

代码示例如下:

5.3 初始化资源

调用初始化【ELTE_SDK_Init(初始化)】接口,初始化SDK资源以及预分配内存等操作。

□ 说明

在使用SDK提供的相关功能之前,需要调用此接口初始化相关资源。

入参为1时第三方接受视频流,入参为0时API接受视频流,并且可以调用语音点呼和组呼业务的功能。

代码示例如下:

```
//cpp code
ELTE_INT32 iRet = ELTE_SDK_Init(0);
if(iRet==0)
{
```

//成功

5.4 设置回调函数

eSDK eLTE大部分的业务接口都是异步的,调用接口成功只表明给系统下达指令成功,如果想获取接口的实际返回情况,需要调用【ELTE_EventCallBack(事件回调函数)】接口,解析回调函数中相应事件类型的消息体。

```
//capp code
//call back function declaration
static ELTE_VOID __SDK_CALL ELTE_EventCallBack(ELTE_INT32 iEventType, ELTE_VOID* pEventBuf,
ELTE_UINT32 uiBufSize, ELTE_VOID* pUserData);
//set call back function
ELTE_INT32 iRet = ELTE_SDK_SetEventCallBack(ELTE_EventCallBack, NULL);
if(iRet==0)
{
    //成功
}
```

5.5 用户登录 eAPP 系统

调用用户登录【ELTE_SDK_Login(用户登录)】接口,登录eAPP系统。您需要填入用户名、密码、调度机IP地址、与调度机通信的本地IP地址、调度机Sip端口号。

代码示例如下:

```
//cpp code
ELTE_INT32 iRet =ELTE_SDK_Login("4120", "4120", "172.22.9.105", "172.24.4.253", 5060);
if(iRet==0)
{
    //成功
}
```

5.6 触发状态上报

在调用触发状态上报接口之前,必须收到自动下载配置

ELTE_Event_NotifyProvisionAllResync完成的回调消息,所有的业务接口都要等收到该回调消息之后才可以调用。

调用触发状态上报【ELTE_SDK_TriggerStatusReport(触发状态上报)】接口,开启状态上报。当用户上下线时,应用程序会收到相应的状态回调消息。

代码样例如下:

```
//cpp code
ELTE_INT32 iRet = ELTE_SDK_TriggerStatusReport(1);
if(iRet==0)
{
    //接口调用成功
}
```

6 典型业务开发场景

本章的任务是介绍典型开发场景,包括场景介绍、接口调用流程、接口调用前提条件、接口调用入参和输出等。

本章按照如下五种典型开发场景进行介绍:

- 6.1 资源管理
- 6.2 语音调度
- 6.3 视频调度
- 6.4 短数据
- 6.5 订阅终端GIS

6.1 资源管理

6.1.1 概述

华为eSDK eLTE SDK提供资源管理的开放接口,用户管理和群组管理包括资源用户和普通群组的基本信息查询接口,增删改功能在eAPP配套的运营支撑系统上进行操作,详情请访问eLTE集群运营支撑系统产品文档。动态群组和派接组支持增删查改功能接口,临时群组支持增查功能接口。

□说明

对于用户和普通群组的增删改功能只能在在eAPP配套的运营支撑系统上进行操作,详情请访问 eLTE集群运营支撑系统产品文档。

eAPP产品支持按照一定条件自动对资源的语音和视频通话进行录制保存,此配置需要在调度机上预先配置,详情请访问eAPP产品文档eAPP产品文档。eSDK eLTE OCX支持查询录音录像文件列表接口,并且通过调度员用户鉴权后可以回放录音录像文件。

□ 说明

eAPP产品中如何配置自动录音录像策略,请参考eAPP产品文档相关章节。

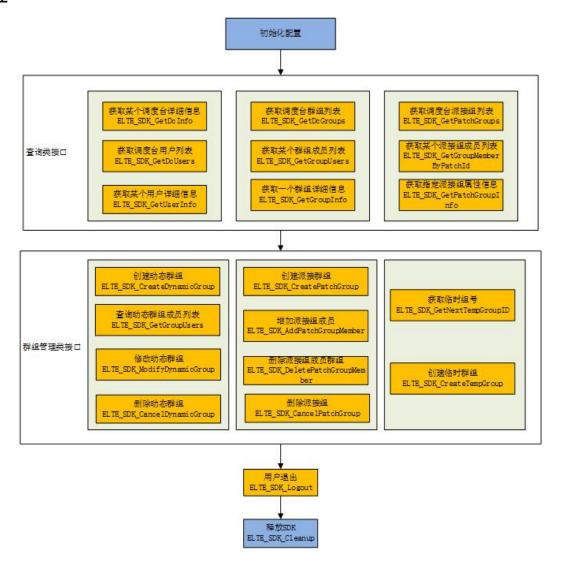
我们将按照如下场景进行详细的分析和介绍:

- 用户和群组管理
- 获取录音录像文件列表

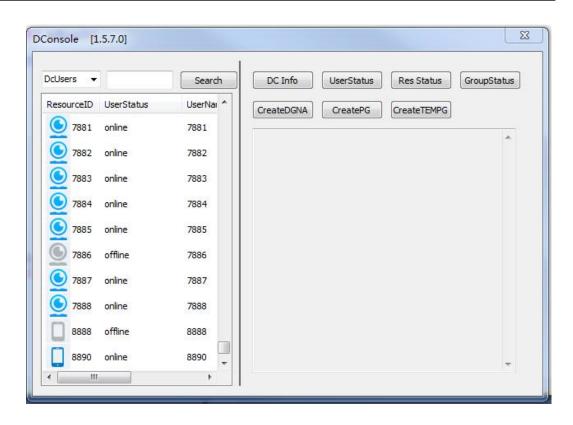
6.1.2 用户和群组管理

本章节的开发任务: 获取调度台信息, 获取调度台用户列表, 获取一个用户的详细信息, 获取调度台群组列表, 获取某个群组成员列表, 获取某个群组的详细信息, 创建动态群组, 获取动态群组成员列表, 删除动态群组。

接口调用流程



用户和群组管理Sample界面:



前提条件

己完成初始化配置。

获取调度台信息

调用【ELTE_SDK_GetDcInfo(获取某个调度台详细信息)】接口,获取DC调度台信息,接口入参是调度台用户ID和一个空指针,空指针带回xml报文包含调度台ID、别名、角色、权限等信息。其他的查询类接口同样需要先定义一个空指针。

□说明

只能调用此接口查询当前登录的调度台信息。

接口调用代码示例如下:

```
//cpp code
//定义一个字符型空指针
ELTE_CHAR* pChar = NULL;
//调用获取当前登录调度台4116的详细信息
ELTE_INT32 iRet = ELTE_SDK_GetUserInfo("4116", &pChar);
//判断接口调用是否成功,成功后释放字符型指针的内存
if(0 == iRet)
{
    //接口调用成功,可通过消息框或其他方式显示查询到的结果
    MessageBox(pChar);
    //释放字符型指针pChar的内存
    ELTE_SDK_ReleaseBuffer(pChar);
}
```

pChar字符型指针返回的调度台信息如下:

```
//xml code

<Content>

<DcInfo>

<DcID>4116</DcID>

<Priviledge>65535</Priviledge>
```

DcInfo节点包含该调度台的DcID、Priviledge、Role、Alias信息,节点说明请参考接口文档中【ELTE_SDK_GetDcInfo(获取某个调度台详细信息)】,您可根据开发需求对这些信息进行处理和显示。

获取调度台用户列表

调用【ELTE_SDK_GetDcUsers(获取调度台用户列表)】接口,获取DC调度台可以管理的所有用户列表,接口返回的xml报文包含UserID、UserCategory、UserPriority、UserName。

接口调用代码示例如下。

```
//cpp code
//定义一个字符型空指针
ELTE_CHAR* pChar = NULL;
//获取调度台的用户列表, 4116是调度台ID, &PChar是返回的用户列表xml报文
ELTE_INT32 iRet = ELTE_SDK_GetDcUsers("4116", &pChar);
//判定接口调用是否成功
if(0 == iRet)
{
    //成功,释放申请的内存
    ELTE_SDK_ReleaseBuffer(pChar);
}
```

pChar字符型指针返回调度台用户列表信息如下:

```
//xml code
<Content>
   <UserInfoList>
       (UserInfo)
            <UserID>4116</UserID>
            <UserCategory>0</UserCategory>
            <UserPriority>15</UserPriority>
            <UserName>4116</UserName>
       </UserInfo>
       <UserInfo>
            <UserID>8895</UserID>
            <UserCategory>9</UserCategory>
            <UserPriority>15</UserPriority>
            <UserName>8895</UserName>
       </UserInfo>
   </UserInfoList>
```

UserInfoList节点包含UserID、UserCategory、UserPriority、UserName信息,节点说明请参考接口文档中【ELTE_SDK_GetDcUsers(获取调度台用户列表)】,用户类别UserCategory的值不同代表不同类型用户,如下。

- 0: Dispatcher, 调度台用户
- 1: FIXEDCAMERA, 固定摄像头
- 2: PSTNUSER, PSTN用户
- 3: TETRAUSER, TETRA用户
- 4: PLMNUSER, PLMN移动用户
- 5: EXTERNALPTT, 外部PTT用户

- 6: SDKUSER, SDK网关内部用户
- 7: APPUSER, 公网APP终端用户
- 8: UELOGINUSER, 终端登录用户
- 9: PTTUSER, PTT用户
- 50: ALLTYPEUSER, 所有用户, 不分类型
- 100: DECUSER, DEC用户
- 255: OTHERUSER, 其它未分类用户

后续您可按照自己的需求分类显示用户列表或用户图标等。

获取某个用户详细信息

调用【ELTE_SDK_GetUserInfo(获取某个用户详细信息)】接口,获取某个用户的详细信息,接口返回的xml报文包含UserID、UserCategory、UserPriority、UserName。

接口调用代码示例如下。

```
//cpp code
//定义一个字符型空指针
ELTE_CHAR* pChar = NULL;
//获取某个用户详细信息,8895是待获取的用户ID,&PChar是返回的用户信息xml报文
ELTE_INT32 iRet = ELTE_SDK_GetUserInfo("8895", &pChar);
//判定接口调用是否成功
if(0 == iRet)
{
    //接口调用成功,释放申请的内存
    ELTE_SDK_ReleaseBuffer(pChar);
}
```

pChar字符型指针返回的用户信息如下:

UserInfo节点说明请参考接口文档中【ELTE_SDK_GetUserInfo(获取某个用户详细信息)】。

获取调度台群组列表

调用【ELTE_SDK_GetDcGroups(获取调度台群组列表)】接口,获取DC调度台可以管理的所有群组列表,接口返回的xml报文包含GroupID、GroupCategory、GroupPriority、GroupName。

接口调用代码示例如下:

```
//cpp code
//定义一个字符型空指针
ELTE_CHAR* pChar = NULL;
//获取调度台的群组列表, 4116是调度台ID, &PChar是返回的群组列表xml报文
ELTE_INT32 iRet = ELTE_SDK_GetDcGroups("4116", &pChar);
//判定接口调用是否成功
if(0 == iRet)
```

```
{
    //接口调用成功,释放申请的内存
    ELTE_SDK_ReleaseBuffer(pChar);
}
```

pChar字符型指针返回的群组列表如下:

```
//xml code
<Content>
   <GroupInfoList>
       <GroupInfo>
            <GroupID>1001</GroupID>
            <GroupCategory>1</GroupCategory>
            <GroupPriority>15</GroupPriority>
            <GroupName>测试群组</GroupName>
       </GroupInfo>
       <GroupInfo>
            <GroupID>1002</GroupID>
            <GroupCategory>1</GroupCategory>
            <GroupPriority>15</GroupPriority>
            〈GroupName〉测试群组1002〈/GroupName〉
       </GroupInfo>
   </GroupInfoList>
</Content>
```

GroupInfoList节点包含GroupID、GroupCategory、GroupPriority、GroupName信息,节点说明请参考接口文档中【ELTE_SDK_GetDcGroups(获取调度台群组列表)】,其中群组类别GroupCategory如下定义:

- 0: GRP ALLBROADCAST, 全网广播组
- 1: GRP_GENERAL, 普通组
- 2: GRP AREABROADCAST, 区域广播组
- 8: GRP_EXTERNAL, 外部组
- 9: GRP DYNAMICGRP, 动态组
- 10: GRP_ALLTYPE, 所有组

后续您可按照自己的需求分类显示用户列表或用户图标等。

获取某个群组成员列表

调用【ELTE_SDK_GetGroupUsers(获取某个群组成员列表)】接口,获取某个群组成员的详细信息,接口返回的xml报文包含UserID、GroupID、UserPriority、MemberType。

接口调用代码示例如下:

```
//cpp code
//定义一个字符型空指针
ELTE_CHAR* pChar = NULL;
//获取群组1001的成员列表, 1001是群组ID, &PChar是返回的群组成员列表xml报文
ELTE_INT32 iRet = ELTE_SDK_GetGroupUsers("1001", &pChar);
//判定接口调用是否成功
if(0 == iRet)
{
    //接口调用成功,释放申请的内存
    ELTE_SDK_ReleaseBuffer(pChar);
}
```

pChar字符型指针返回的群组成员列表信息如下:

```
//xml code
<Content>
   <GroupUserInfoList>
       <GroupUserInfo>
           <UserID>8888</UserID>
           <GroupID>1001</GroupID>
           <UserPriorityInGroup>15</UserPriorityInGroup>
           <MemberType>1
       </GroupUserInfo>
       <GroupUserInfo>
           <UserID>8890</UserID>
           <GroupID>1001</GroupID>
           <UserPriorityInGroup>15</UserPriorityInGroup>
           <MemberType>1
       </GroupUserInfo>
   </GroupUserInfoList>
</Content>
```

返回信息中各节点说明请参考接口文档中【ELTE_SDK_GetGroupUsers(获取某个群组成员列表)】。

获取某个群组详细信息

调用【ELTE_SDK_GetGroupInfo(获取某个群组详细信息)】接口,获取某个群组的详细信息,接口返回的xml报文包含GroupID、GroupCategory、GroupPriority、GroupName、GroupCreator。

接口调用代码示例如下。

```
//cpp code
//定义一个字符型空指针
ELTE_CHAR* pChar = NULL;
//获取群组1001的详细信息, 1001是待获取的群组ID, &PChar是返回的群组信息xml报文
ELTE_INT32 iRet = ELTE_SDK_GetGroupInfo("1001", &pChar);
//判定接口调用是否成功
if(0 == iRet)
{
    //接口调用成功,释放申请的内存
    ELTE_SDK_ReleaseBuffer(pChar);
}
```

pChar字符型指针返回的群组信息如下:

返回信息中各节点说明请参考接口文档中【ELTE_SDK_GetGroupInfo(获取某个群组详细信息)】。

获取某个调度台的派接组列表

调用【ELTE_SDK_GetPatchGroups(获取某个调度台的派接组列表)】接口,获取指定调度台的所有派接组列表。

∭说明

若使用完该接口,需要调用【ELTE SDK ReleaseBuffer(释放内存)】接口将其资源释放掉。

接口调用代码示例如下:

```
//cpp code
//定义一个字符型空指针
ELTE_CHAR* pChar = NULL;
//获取调度台4116的所有派接组列表,&PChar是返回的派接组信息xml报文
ELTE_INT32 iRet = ELTE_SDK_GetPatchGroups("4116", &pChar);
//判定接口调用是否成功
if(0 == iRet)
{
    //接口调用成功,释放申请的内存
    ELTE_SDK_ReleaseBuffer(pChar);
}
```

pChar字符型指针返回信息格式如下:

```
//xml code
<Content>
   <PatchGroupInfoList>
       <PatchGroupInfo>
           <GroupNumber>99899983</GroupNumber>
           <SetupDcId>4116</SetupDcId>
           <PGPriority>1</PGPriority>
           <DcPatchIndex>1
           <PGName>test1</PGName>
           <VPNID>0</VPNID>
       </PatchGroupInfo>
       <PatchGroupInfo>
           <GroupNumber>99899999</GroupNumber>
           <SetupDcId>4130</SetupDcId>
           <PGPriority>3</PGPriority>
           <DcPatchIndex>1
           <PGName>a1</PGName>
           <VPNID>0</VPNID>
       </PatchGroupInfo>
   </PatchGroupInfoList>
```

返回信息中各节点说明请参考接口文档中【ELTE_SDK_GetPatchGroups(获取某个调度台的派接组列表)】。

获取某个派接组的成员列表

调用【ELTE_SDK_GetGroupMemberByPatchId(获取某个派接组的成员列表)】接口,获取指定派接组管理的成员列表。

门设明

若使用完该接口,需要调用【ELTE_SDK_ReleaseBuffer(释放内存)】接口将其资源释放掉。

```
//cpp code
//定义一个字符型空指针
ELTE_CHAR* pChar = NULL;
//获取派接组9899999的成员列表, &PChar是返回的派接组成员列表信息xml报文
ELTE_INT32 iRet = ELTE_SDK_GetGroupMemberByPatchId("99899999", &pChar);
//判定接口调用是否成功
if(0 == iRet)
{
    //接口调用成功,释放申请的内存
    ELTE_SDK_ReleaseBuffer(pChar);
}
```

pChar字符型指针返回信息格式如下:

```
//xml code

<Content>

<PatchGroupMemberList>

<PatchGroupMember>

<GroupNumber>99899999</GroupNumber>

<MemberGroupId>1001</MemberGroupId>

</PatchGroupMember>

......

</PatchGroupMemberList>

</Content>
```

返回信息中各节点说明请参考接口文档中

【ELTE_SDK_GetGroupMemberByPatchId(获取某个派接组的成员列表)】。

获取某个派接组详细信息

调用【ELTE_SDK_GetPatchGroupInfo(获取某个派接组详细信息)】接口,获取指定派接组属性信息。

□ 说明

若使用完该接口,需要调用【ELTE_SDK_ReleaseBuffer(释放内存)】接口将其资源释放掉。

接口调用代码示例如下:

```
//cpp code
//定义一个字符型空指针
ELTE_CHAR* pChar = NULL;
//获取派接组9899999的详细信息,&PChar是返回的派接组详细信息xml报文
ELTE_INT32 iRet = ELTE_SDK_GetPatchGroupInfo("99899999", &pChar);
//判定接口调用是否成功
if(0 == iRet)
{
    //接口调用成功,释放申请的内存
    ELTE_SDK_ReleaseBuffer(pChar);
}
```

pChar字符型指针返回信息格式如下:

返回信息中各节点说明请参考接口文档中【ELTE_SDK_GetPatchGroupInfo(获取某个派接组详细信息)】。

创建动态群组

调用【ELTE_SDK_CreateDynamicGroup(创建动态群组)】接口,创建动态群组入参包括群组号、创建者(调度机用户ID)、群组成员列表、用户成员列表等信息。

```
//cpp code
//创建一个动态群组
CString strDGNAParam;
```

```
strDGNAParam. Append("<Content>");
//动态群组ID, 可以不填, 由系统自动生成
strDGNAParam. Append("<GroupID>");
strDGNAParam. Append("</GroupID>");
//创建该动态组的DC用户号, 此参数目前不使用, 不用填写
strDGNAParam. Append("<DcID>");
strDGNAParam. Append("</DcID>");
//动态群组别名,可不填,字符串长度小于等于32
strDGNAParam.Append("<Alias>");
strDGNAParam. Append("Test1");
strDGNAParam.Append("</Alias>");
//动态组优先级,取值范围: 1~15
strDGNAParam. Append("<Priority>");
strDGNAParam. Append("15");
strDGNAParam. Append ("</Priority>");
//动态组的最大通话时长,取值范围: 1~66535
strDGNAParam. Append("<MaxPeriod>");
strDGNAParam.Append("60");
strDGNAParam.Append("</MaxPeriod>");
//动态群组列表,最大值为8个,分别传入组ID,动态群组中群组成员列表和用户成员列表可以有一个是空的。
{\tt strDGNAParam.\,Append(`` \langle GroupList \rangle'');}
strDGNAParam. Append("<GroupID>");
strDGNAParam. Append ("1001");
{\tt strDGNAParam.\,Append}\,(\text{``} \text{</GroupID>''})\;;
strDGNAParam. Append("</GroupList>");
//用户成员列表,最大值为200个,分别传入用户ID
strDGNAParam.Append("<UserList>");
strDGNAParam.Append("<UserID>");
strDGNAParam. Append ("8895");
strDGNAParam.Append("</UserID>");
strDGNAParam.Append("</UserList>");
strDGNAParam. Append("</Content>");
//调用创建动态群组接口
ELTE_INT32 iRet = ELTE_SDK_CreateDynamicGroup(strDGNAParam);
//判断接口调用是否成功
if(0 == iRet)
    //成功
```

解析事件回调函数中【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】,动态群组创建成功后回调函数返回消息xml如下:

动态群组创建成功后会自动订阅并加入群组,ResourceID是动态群组ID, StatusValue=4003表示资源指派状态,说明动态群组创建并订阅成功。

获取动态群组成员列表

调用【ELTE_SDK_GetGroupUsers(获取某个群组成员列表)】接口可获取动态群组成员列表,参见获取某个群组成员列表。

修改动态群组

调用【ELTE_SDK_ModifyDynamicGroup(修改动态群组)】接口,修改动态群组,用于向动态组增加用户或者删除用户。

□ 说明

- 1. 在调用该接口之前,确保目标动态群组的创建者为当前登录的调度台用记且处于被订阅状态,订阅群组请调用【ELTE SDK SubJoinGroup(订阅并加入群组)】。
- 2. 调用该接口前,可通过调用【获取某个群组成员列表】接口,从该列表中获取待修改的动态 群组成员信息。

接口调用代码示例如下:

```
//cpp code
CString strDGNAParam:
strDGNAParam. Append("<Content>");
strDGNAParam.Append("<DynamicGroupInfo>");
//动态群组ID,必填
strDGNAParam. Append("<GroupID>");
strDGNAParam. Append("99899964");
strDGNAParam. Append("</GroupID>");
//待添加到动态群组的用户成员ID,增加或删除成员不能都不填写
strDGNAParam. Append ("<AddUserList>");
strDGNAParam. Append ("<AddUserID>");
strDGNAParam. Append ("</AddUserID>");
strDGNAParam. Append ("</AddUserList>");
//待删除的动态群组用户成员ID
strDGNAParam. Append ("<DeleteUserList>"); strDGNAParam. Append ("<DeleteUserID>");
strDGNAParam.Append("8895");
strDGNAParam.Append("</DeleteUserID>");
strDGNAParam.Append("</DeleteUserList>");
strDGNAParam. Append ("</DynamicGroupInfo>");
strDGNAParam. Append("</Content>");
//调用修改动态群组接口,第一个入参是群组的创建者,只有创建者也是当前登录的调度台才能执行成功
ELTE_INT32 iRet = ELTE_SDK_ModifyDynamicGroup("4116", strDGNAParam);
//判断接口调用是否成功
if(iRet==0)
     //接口调用成功
```

解析事件回调函数中【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】,动态群组修改成功后回调函数返回消息xml如下:

StatusType表示用户动态重组状态,StatusValue=4024表示资源动态重组成功,说明动态群组修改成功。

删除动态群组

调用【ELTE_SDK_CancelDynamicGroup(删除动态群组)】接口,删除动态群组入参是动态组号。



注意

- 1. 只有订阅并加入该动态群组后,并且是该动态群组的创建者才能成功删除它。
- 2. 删除动态群组之前动态群组必须处于已经订阅的状态。

接口调用代码示例如下:

```
//cpp code
//删除动态群组
ELTE_INT32 iRet = ELTE_SDK_CancelDynamicGroup("99899964");
if(iRet==0)
{
//成功
}
```

解析事件回调函数中【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】,动态群组删除成功后回调函数返回消息xml如下:

ResourceID=动态群组ID, StatusValue=4004资源去指派状态,说明动态群组删除成功。 未订阅并加入群组返回iRet返回-40011错误码说明用户未订阅并加入群组删除失败;非 动态群组的创建者删除派接组iRet返回-40005错误码说明用户无权限删除失败。

创建临时群组

调用【ELTE_SDK_CreateTempGroup(创建临时群组)】接口创建临时群组,临时群组没有对应的删除接口,在用户注销或组呼业务完成后或者长时间没有操作临时群组会自动消失。

```
//cpp code
//先定义一个整型的临时群组ID
ELTE INT32 TempGroupId;
//调用获取临时群组号接口,输出的参数即临时群组ID
ELTE_INT32 iRet = ELTE_SDK_GetTempGroupID(&TempGroupId);
-----
//把整型TempGroupId转化为字符串strGroupID
char strGroupID[256] = {'\0'};
sprintf_s(strGroupID, "%d", TempGroupId);
//构造一个创建临时群组的入参字符串
CString strMsg;
strMsg. Append("<Content>");
//临时群组ID,通过上面接口获取
strMsg. Append("<GroupID>");
strMsg.Append(strGroupID);
strMsg.Append("</GroupID>");
//临时群组的创建者,调度台ID,可不填
strMsg. Append("<DcID>");
strMsg. Append("</DcID>");
//别名,可不填写
strMsg. Append("<Alias>");
strMsg. Append("</Alias>");
//群组优先级,可不填写
strMsg.Append("<Priority>");
strMsg. Append("</Priority>");
```

```
//最大通话时长,可不填
strMsg.Append("<MaxPeriod>");
strMsg. Append ("</MaxPeriod>");
//临时群组中的群组成员,最大值为8个,分别传入组ID,只要群组成员和用户成员不是都空即可
strMsg.Append("\GroupList\");
strMsg.Append("\GroupID\");
strMsg. Append ("</GroupID>");
strMsg. Append ("</GroupList>");
strMsg. Append ("</GroupList>");
//临时群组的用户成员列表,最大值为200个,分别传入用户ID
strMsg.Append("<UserList>");
strMsg. Append ("<UserID>");
strMsg. Append ("8894");
strMsg. Append("</UserID>");
strMsg. Append("<UserID>");
strMsg. Append("8895");
strMsg. Append("</UserID>");
strMsg. Append("</UserList>");
strMsg. Append("</Content>");
//调用创建临时群组的接口
iRet = ELTE SDK CreateTempGroup(strMsg);
//若接口调用失败,输出错误码
if(0 != iRet)
     char strResult[256] = {' \setminus 0'};
     sprintf s(strResult, "%d", iRet);
    MessageBox(strResult);
```

解析事件回调函数中【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】,临时群组创建成功后回调函数返回消息xml如下:

临时群组创建成功后会自动订阅并加入群组,ResourceID是临时群组ID,StatusValue=4003表示资源指派状态,说明临时群组创建并订阅成功。

创建派接组

调用【ELTE SDK CreatePatchGroup(创建派接组)】接口创建派接组。

□ 说明

- 1. 当群组A中的某一用户L(用户L必须是调度台用户)发起派接到群组B,用户L发起组呼通话,则群组B也能接收到L的组呼通话,即使L不是群组B的成员。
- 2. 目前派接组成员只支持普通群组。

```
//cpp code
//构造一个创建派接组入参xml字符串
CString strPatchGroupParam;
strPatchGroupParam. Append("<Content>");
//创建派接组的调度员ID,可不填写
strPatchGroupParam. Append("<DcID>");
strPatchGroupParam. Append("</DcID>");
//派接组的组号,首次创建的派接组组号默认设置为0,以后系统为此派接组自动分配一个派接组号。
strPatchGroupParam. Append("<PatchGroupID>");
strPatchGroupParam. Append("</PatchGroupID>");
//派接组名称
strPatchGroupParam. Append("<PatchGroupName>");
```

```
strPatchGroupParam. Append("PatchTestGroup");
strPatchGroupParam. Append("</PatchGroupName>");
//派接组成员列表,最多支持20个普通组被派接
strPatchGroupParam. Append("<PatchGroupMemberList>");
strPatchGroupParam. Append("<PatchGroupMember>");
strPatchGroupParam. Append("<PatchGroupMember>");
strPatchGroupParam. Append("</PatchGroupMember>");
strPatchGroupParam. Append("</PatchGroupMemberList>");
strPatchGroupParam. Append("</Content>");
//调用创建派接组接口
ELTE_INT32 iRet = ELTE_SDK_CreatePatchGroup(strPatchGroupParam);
//若接口调用失败,输出错误码
if(0 != iRet)
{
    char strResult[256] = {'\0'};
    sprintf_s(strResult, "%d", iRet);
    MessageBox(strResult);
}
```

解析事件回调函数中【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】,派接组创建成功后回调函数返回消息xml如下:

```
//xml code

<Content>

<ResourceID>99899959</ResourceID>

<ResourceName></ResourceName>

<StatusType>24</StatusType>

<StatusValue>4031</StatusValue>

<AttachingGroup>0</AttachingGroup>

<Cause>0</Cause>

<MemberID>0</MemberID>

</Content>
```

ResourceID是派接组ID,StatusType=24表示派接操作状态,StatusValue=4031表示派接组创建成功。

增加派接组成员

调用【ELTE_SDK_AddPatchGroupMember(增加派接组成员)】接口,调度员增加派接组成员,该成员是一个普通群组。

□ 说明

仅派接组的创建者有权限增加派接组成员。

```
//cpp code
//构造一个增加派接组的入参xml字符串
CString strPatchGroupParam;
strPatchGroupParam. Append("<Content>");
//派接组的创建者调度员ID, 可不填写
strPatchGroupParam.Append("<DcID>");
strPatchGroupParam. Append("</DcID>");
//派接组ID
strPatchGroupParam.\ Append\ (\text{```<} PatchGroupID>'')\ ;
strPatchGroupParam. Append("99899959");
strPatchGroupParam.Append("</PatchGroupID>");
//派接组名称
strPatchGroupParam. Append("<PatchGroupName>");
strPatchGroupParam.Append("PatchTestGroup");
strPatchGroupParam. Append("</PatchGroupName>");
//派接组成员列表,最多支持20个普通组被派接
strPatchGroupParam.Append("<PatchGroupMemberList>");
strPatchGroupParam. Append ("<PatchGroupMember>"); strPatchGroupParam. Append ("1001"); strPatchGroupParam. Append ("</PatchGroupMember>");
strPatchGroupParam. Append("</PatchGroupMemberList>");
strPatchGroupParam. Append("</Content>");
```

```
//调用增加派接组成员接口
ELTE_INT32    iRet = ELTE_SDK_AddPatchGroupMember(strPatchGroupParam);
//若接口调用失败,输出错误码
if(0 != iRet)
{
    char strResult[256] = {'\0'};
    sprintf_s(strResult, "%d", iRet);
    MessageBox(strResult);
}
```

解析事件回调函数中【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】,增加派接组成员成功后回调函数返回消息xml如下:

ResourceID是派接组ID, StatusType=24表示派接操作状态, StatusValue=4035表示派接组增加成员成功, MemberID=1001表示增加的派接组成员是1001群组。

删除派接组成员

调用【ELTE_SDK_DeletePatchGroupMember(删除派接组成员)】接口,派接组的创建者调度员才能调用该接口,删除组内成员。

接口调用代码示例如下:

```
//cpp code
//构造一个删除派接组的入参xml字符串
CString strPatchGroupParam;
strPatchGroupParam. Append("<Content>");
//派接组的创建者调度员ID, 可不填写
strPatchGroupParam.Append("<DcID>");
strPatchGroupParam.Append("</DcID>");
//派接组ID
strPatchGroupParam.Append("<PatchGroupID>");
strPatchGroupParam.Append("99899959");
strPatchGroupParam.Append("</PatchGroupID>");
//派接组别名
strPatchGroupParam. Append("<PatchGroupName>");
strPatchGroupParam. Append("PatchTestGroup");
strPatchGroupParam. Append("//PatchGroupName>");
//待删除的派接组成员列表,最多支持20个普通组被派接
strPatchGroupParam.\ Append (``\langle PatchGroupMemberList\rangle'');
strPatchGroupParam.\ Append(``\langle PatchGroupMember\rangle'');\\ strPatchGroupParam.\ Append("1001");
strPatchGroupParam. Append("</PatchGroupMember>");
strPatchGroupParam. Append("</PatchGroupMemberList>"); strPatchGroupParam. Append("</Content>");
//调用删除派接组成员列表
ELTE_INT32 iRet = ELTE_SDK_DeletePatchGroupMember(strPatchGroupParam);
//若接口调用失败,输出错误码
if(0 != iRet)
    char strResult[256] = {' \setminus 0'};
    sprintf_s(strResult, "%d", iRet);
    MessageBox(strResult);
```

解析事件回调函数中【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】,删除派接组成员成功后回调函数返回消息xml如下:

ResourceID是派接组ID, StatusType=24表示派接操作状态, StatusValue=4037表示派接组删除成员成功, MemberID=1001表示删除的派接组成员是1001群组。

取消派接组

调用【ELTE_SDK_CancelPatchGroup(取消派接组)】接口,派接组的创建者调度员才有权调用此接口取消派接组。

接口调用代码示例如下:

```
//cpp code

//调用取消派接组接口

ELTE_INT32 iRet = ELTE_SDK_CancelPatchGroup("99899959");

//若接口调用失败,输出错误码

if(0 != iRet)

{

    char strResult[256] = {'\0'};

    sprintf_s(strResult, "%d", iRet);

    MessageBox(strResult);

}
```

解析事件回调函数中【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】,取消派接组成功后回调函数返回消息xml如下:

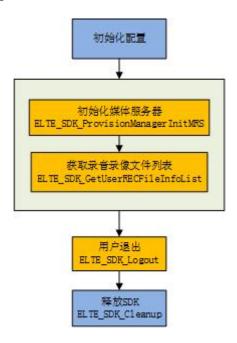
ResourceID是派接组ID,StatusType=24表示派接操作状态,StatusValue=4033表示派接组取消创建成功。

6.1.3 获取录音录像文件列表

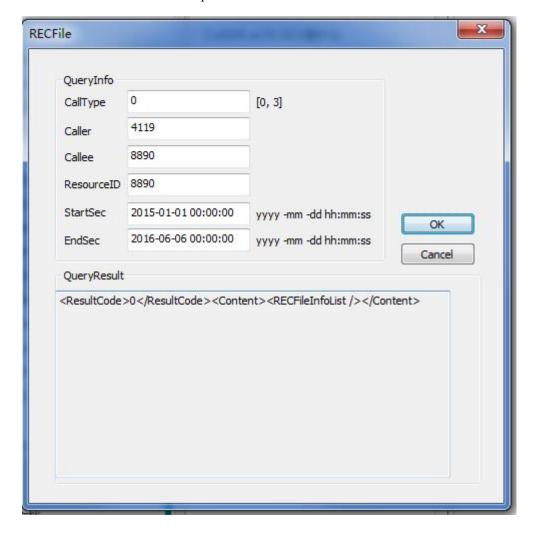
本章节的开发任务是介绍怎么获取录音录像文件列表,包括调用流程、调用方法、注意事项等。

在调用获取录音录像文件之前,首先要调用初始化媒体服务器接口。

接口调用流程



获取录音录像文件列表的Sample界面:



前提条件

- 1. 已完成初始化配置。
- 2. MRS系统设置了录播计划,调度台有查询录音录像文件的权限。

初始化媒体服务器

调用【ELTE_SDK_ProvisionManagerInitMRS(初始化媒体服务器)】接口初始化媒体服务器,媒体服务器在单站场景中与eMDC的IP相同,在分布式组网中与eMDC是不同的ip地址,在调用该接口前要给用户一个可以输入媒体服务器IP的页面入口。

接口调用代码样例如下:

```
//cpp code
//初始化媒体服务器
ELTE_INT32 iRet = ELTE_SDK_ProvisionManagerInitMRS("172.22.9.105");
```

接口调用后返回0,说明初始化媒体服务器成功,-999表示连接媒体服务器失败,可能是媒体服务器IP地址填写错误。其他的返回值请参考**错误码**。

获取录音录像文件列表

调用【ELTE_SDK_GetUserRECFileInfoList(获取录音录像文件列表)】获取录音录像文件列表,入参包括呼叫类型、主叫号码、被叫号码、资源ID、时间段和资源ID等查询条件。



注意

- 1. 调用该接口前,确保已调用ELTE_SDK_ProvisionManagerInitMRS(初始化媒体服务器)接口并且返回成功。
- 2. 调用完该接口,需要调用 ELTE_SDK_ReleaseBuffer (释放内存)接口将其资源释放掉。
- 3. 从安全角度考虑,查询到的录音录像文件列表不要保存在日志文件中。

```
//cpp code
//定义查询条件字符串
CString strQuery;
strQuery.Append("<Content>");
strQuery.Append("<RECQueryInfo>");
//呼叫类型: 0: 语音点呼
                         1: 视频点呼
                                        2: 视频回传
                                                      3: 群组呼叫
strQuery. Append("<CallType>");
strQuery. Append("0");
strQuery.Append("</CallType>");
//主叫号码,默认值可填-1, 如果Caller填主叫号码,Callee和ResourceID可填-1;
                                                                        如果Caller填主叫号码
且Callee填被叫号码, ResourceID可填-1;
strQuery.Append("<Caller>");
strQuery.Append("8890");
strQuery. Append("</Caller>");
//被叫号码,默认值可填-1,如果Callee填被叫号码,则Caller和ResourceID可填-1; 如果Callee填被叫号码
且Caller填主叫号码,则ResourceID可填-1;
strQuery. Append("<Callee>");
strQuery. Append("-1");
strQuery. Append("</Callee>");
//资源ID(包括群组ID和终端ID),默认值可填-1,如果CallType填0或1,caller和callee都填-1,则ResourceID
填主叫或者被叫的号码;如果CallType填2,caller和callee都填-1,则ResourceID填录音录像文件的资源ID;如
```

```
果CallType填3, caller和callee都填-1,则ResourceID填群组号或终端号;
strQuery.Append("<ResourceID>");
strQuery. Append("-1");
strQuery. Append ("</ResourceID>");
//查询开始时间,格式如: yyyy-mm-dd hh:mm:ss, 不填写表示查询所有时间
strQuery.Append("<StartSec>");
strQuery.Append("</StartSec>");
//查询结束时间,格式如: yyyy-mm-dd hh:mm:ss, 不填写表示查询所有时间strQuery.Append("<EndSec>");
strQuery. Append("</EndSec>");
strQuery.Append("</RECQueryInfo>");
strQuery.Append("</Content>");
//定义一个输出参数
ELTE_CHAR* pQueryResult = NULL;
//调用接口
ELTE_INT32 iRet = ELTE_SDK_GetUserRECFileInfoList(strQuery, &pQueryResult);
if(0 == iRet)
    //接口调用成功后释放内存
   ELTE SDK ReleaseBuffer(pQueryResult);
```

接口调用成功后pQueryResult返回的查询结果如下:

```
//xml code
 <Content>
               <RECFileInfoList>
                              <RECFileInfo>
                                             <CallType>0</CallType>
                                             <Caller>8890</Caller>
                                             <Callee>10086</Callee>
                                             <ResourceID>0</ResourceID>
                                             <StartSec>2016-09-28 21:21:52
                                             <EndSec>2016-09-28 21:22:06</EndSec>
                                             <UrlFTP>http://172.22.9.120:8000/ubp/rec/
2016/09/28/21/8890\_10086\_20160928212152/8890\_10086\_20160928212152\_812940.\ \texttt{amr} \\ \\ < |\text{Ur1FTP}| \\ < |\text{U
                                             <UrlRTSP>rtsp://172.22.9.120:8554/1_84fefd6a-857e-11e6-8000-4cb16cf61a04_1//UrlRTSP>
                               </RECFileInfo>
                              <RECFileInfo>
                                             <CallType>0</CallType>
                                             <Caller>8890</Caller>
                                             <Callee>8894</Callee>
                                             <ResourceID>0</ResourceID>
                                             <StartSec>2016-09-20 19:45:02</StartSec>
                                             <EndSec>2016-09-20 19:50:08</EndSec>
                                             <UrlFTP>http://172.22.9.120:8000/ubp/rec/
</RECFileInfo>
               </RECFileInfoList>
</Content>
```

查询结果中UrlFTP和UrlRTSP通过调度员鉴权后可以直接回放视频。

6.2 语音调度

6.2.1 概述

eSDK eLTE SDK开放语音调度接口,

包括语音点呼、语音组呼、缜密监听和环境监听,是eLTE宽带集群的核心功能之一。

语音点呼功能相关的接口包括发起语音点呼、拒接语音点呼、接受语音点呼、挂断语音点呼、强拆点呼、抢话、发起人工转接。

语音组呼功能相关的接口包括发起组呼或抢权、释放话语权、退出组呼、发起紧急组呼、强拆组呼。

语音点呼、语音组呼以及后面要提到的带伴音的视频回传都可以共用的接口包括执行静音、取消静音。

缜密监听功能接口包括发起缜密监听和停止缜密监听,可对终端、调度台、普通群组和动态群组。

环境监听只能针对终端用户才能使用发起环境监听,结束环境监听的接口使用挂断语音电话接口。

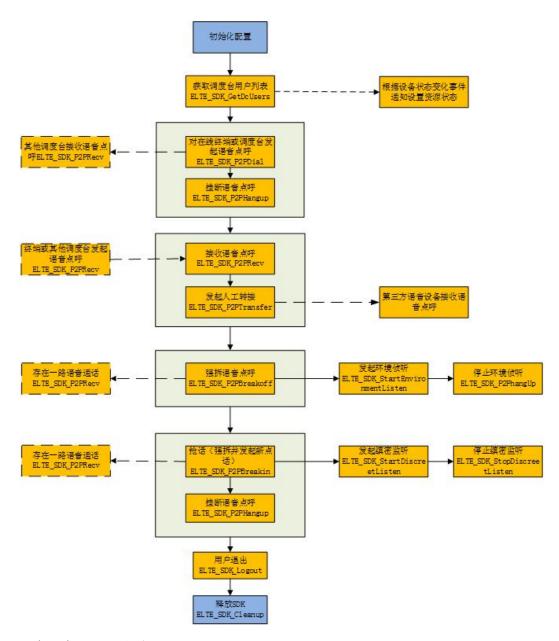
我们将按照如下场景进行详细的分析和介绍:

- 语言点呼场景
- 语言组呼场景

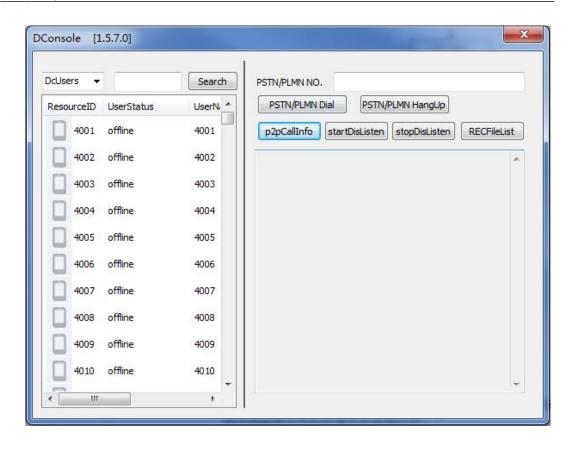
6.2.2 语音点呼场景

本章节的开发任务:调用语音点呼相关接口,进行语音点呼相关操作。

语音点呼接口调用流程:



语音调度Sample界面:

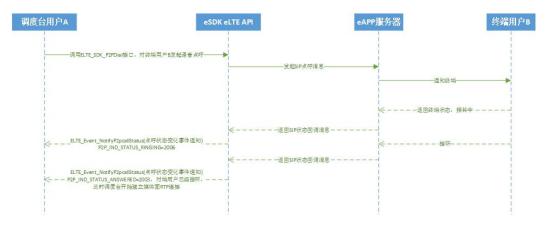


前提条件

己完成初始化配置。

发起语音点呼

调用【ELTE_SDK_P2PDial(发起语音点呼)】接口,向在线的终端、调度台或与eAPP系统对接的其他支持语音点呼的设备发起语音点呼。下面以调度台用户A向终端用户B发起语音点呼为例,调用时序图如下:



```
//cpp code
//对一个集群终端发起语音点呼
ELTE_INT32 iRet = ELTE_SDK_P2PDial("8890");
if(0 == iRet)
```

```
//接口调用成功
```

接口调用成功返回0,语音点呼状态需通过处理回调函数

【ELTE_Event_NotifyP2pcallStatus(点呼状态变化事件通知)】来解析出点呼状态。回调函数返回点呼状态变化事件通知包括语音点呼状态P2pcallStatus、主叫Caller、被叫Calleee、抢权者Inserter、被强权者Targeter、音频格式Soundtype等。

回调函数返回示例如下:

```
//xml code
<Content>
   <P2pcal1Status>2005</P2pcal1Status>
   <Caller>4135</Caller>
   <Callee>8890</Callee>
   <Inserter>0</Inserter>
   <Targeter>0</Targeter>
   <Transfer></Transfer>
   <LocalPort>0</LocalPort>
   <RemotePort>0</RemotePort>
   <RemoteIP>0.0.0.0/RemoteIP>
   <SoundPtype>-1</SoundPtype>
   <CallID>0</CallID>
   <SignalError>0</SignalError>
   <FromString>4135
   <ToString>8890</ToString>
```

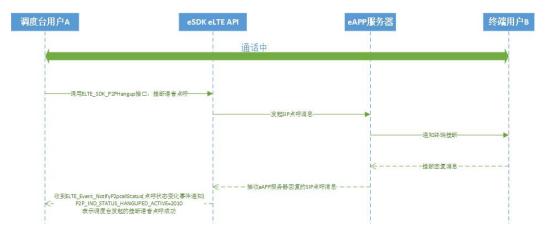
其中P2pcallStatus就代表状态事件,有如下状态:

- P2pcallStatus=2005, 呼叫进行中。
- P2pcallStatus=2006,SDK会自动播放铃声,提示用户可以取消呼叫。
- P2PcallStatus=2003,对端已经接听。

其他状态请参考【ELTE Event NotifyP2pcallStatus(点呼状态变化事件通知)】。

挂断语音点呼

调用【ELTE_SDK_P2PHangup(挂断语音点呼)】接口,调度台挂断与其他设备的语音点呼。下面以调度台用户A挂断与终端用户B之间的语音点呼为例,调用时序图如下。



接口调用代码示例:

```
//cpp code
//挂断调度台与终端8890的语音点呼
ELTE_INT32 iRet = ELTE_SDK_P2PHanpup("8890");
if(0 == iRet)
```

```
{
  //接口调用成功
}
```

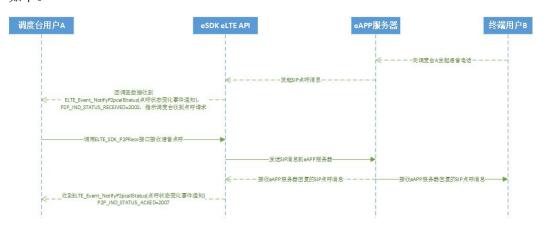
接口调用成功后收到状态事件P2pcallStatus=2010表示调度台主动挂断成功,若收到状态事件P2pcallStatus=2009表示对端已经先挂断。

其他状态请参考【ELTE Event_NotifyP2pcallStatus(点呼状态变化事件通知)】。

接收语音点呼

当调度台收到来自其他用户的语音点呼时,首先是解析回调函数中

【ELTE_Event_NotifyP2pcallStatus(点呼状态变化事件通知)】,如果点呼状态事件P2pcallStatus=2002,则调度台可调用【ELTE_SDK_P2PRecv(接收语音点呼)】接口来接听语音点呼,下面以调度台用户A接收到终端用户B发起语音点呼为例,调用时序图如下。



接听语音点呼接口调用代码示例:

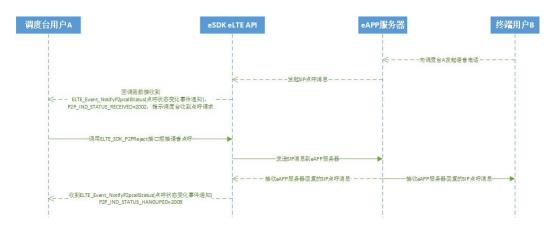
```
//cpp code
//接听来自终端8890的语音点呼
LTE_INT32 iRet = ELTE_SDK_P2PRecv("8890");
if(0 == iRet)
{
    //接口调用成功
}
```

接听接口调用成功后收到状态事件P2pcallStatus=2007表示接听语音点呼成功。

其他状态请参考【ELTE Event NotifyP2pcallStatus(点呼状态变化事件通知)】。

拒接语音点呼

当调度台收到来自其他用户的语音点呼时,首先是解析回调函数中的点呼状态事件,如果点呼状态事件P2pcallStatus=2002,则调度台可调用【ELTE_SDK_P2PReject(拒接语音点呼)】接口来拒接语音点呼,下面以调度台用户A拒接终端B的语音点呼为例,调用时序图如下:



拒接语音点呼接口调用代码示例:

拒接接口调用成功后收到状态事件P2pcallStatus=2009表示拒绝语音点呼成功,或挂断成功,其他状态请参【ELTE_Event_NotifyP2pcallStatus(点呼状态变化事件通知)】。

发起人工转接

调用【ELTE_SDK_P2PTransfer(发起人工转接)】接口发起人工转接,入参需要以对应的XML报文形式传入,入参分别是发起人工转接的调度台号码DCID、当前正在通话中的对端SpeakerID、需要转接的号码ObjectID。下面以调度台用户A与终端用户B正在进行的语音通话转接调度台C,调用时序图如下:



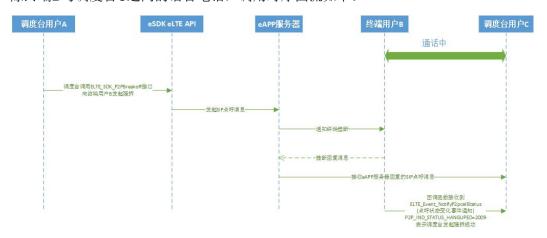
```
//cpp code
//构造一个人工转接入参xml字符串
CString strP2PTransferParam;
```

```
strP2PTransferParam. Append("<Content>");
//发起人工转接的调度员用户ID
strP2PTransferParam.Append("<DcID>");
strP2PTransferParam. Append ("4116");
strP2PTransferParam. Append("</DcID>");
//当前正在通话中的对端的ID
strP2PTransferParam.Append("<SpeakerID>");
strP2PTransferParam. Append("8895");
strP2PTransferParam.Append("</SpeakerID>");
//需要转接的号码ID
strP2PTransferParam.Append("<0bjectID>");
strP2PTransferParam.Append("4135");
strP2PTransferParam. Append ("</ObjectID>");
strP2PTransferParam.Append("</Content>");
ELTE_INT32 iRet = ELTE_SDK_P2PTransfer("4116", strP2PTransferParam);
if(0 == iRet)
   //接口调用成功
```

人工转接接口调用成功后收到点呼状态事件P2pcallStatus=2025表示人工转接成功,其他状态请参考【ELTE_Event_NotifyP2pcallStatus(点呼状态变化事件通知)】。

强拆语音点呼

调用【ELTE_SDK_P2PBreakoff(强拆语音点呼)】表示调度台作为第三方强行拆除某个用户的当前呼叫,被强拆点呼的用户可以是调度台或手持终端。下面以调度台A强行拆除终端B与调度台C之间的语音电话,调用时序图流如下。



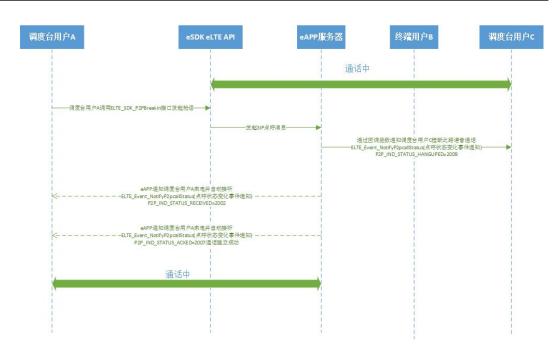
接口调用代码示例如下:

```
//cpp code
//调度台4114强拆终端8890与其他用户的通话
ELTE_INT32 iRet = ELTE_SDK_Brealoff("8890");
if(0 == iRet)
{
    //接口调用成功
}
```

接口调用成功iRet返回0,返回其他错误码可参考【eLTE产品错误码】。

抢话语音点呼

调用【ELTE_SDK_P2PBreakin(抢话)】表示调度台作为第三方强行拆除某个用户的当前呼叫,并收到该用户向调度台发起的新点呼。被抢话的用户可以是调度台或手持终端。下面以调度台用户A抢终端B与调度台用户C之间的语音通话,最终调度台A与终端B建立语音通话,调用时序图如下。



接口调用代码示例如下。

```
//cpp code

//调度台4114对手持终端8890发起抢话

ELTE_INT32 iRet = ELTE_SDK_P2PBreakin("8890");

if(0 == iRet)

{

    //接口调用成功

}
```

接口调用成功Ret返回为0,其他错误码可参考【eLTE产品错误码】。

执行静音

调用【ELTE_SDK_VolMute(执行静音)】接口,实现在点对点呼叫、组呼、视频回传场景中执行静音,目前只支持对通话对端用户静音。

接口调用示代码示例如下:

```
//cpp code
//构造静音参数xml字符串
CString strMuteParam;
strMuteParam.Append("<Content>");
strMuteParam.Append("<MuteParam>");
//呼叫类型:
               0: 点呼
                            1: 组呼
                                         2: 视频回传
strMuteParam. Append("<CallType>");
strMuteParam. Append("0");
strMuteParam. Append("</CallType>");
strMuteParam. Append("</MuteParam");
strMuteParam. Append("</Content>");
//调度台对手持终端8895发起的语音点呼执行静音
ELTE INT32 iRet = ELTE SDK VolMute("8895", strMuteParam);
//判断接口调用是否成功
if(0 == iRet)
     //接口调用成功
```

接口调用成功Ret返回为0,其他错误码可参考【eLTE产品错误码】。

取消静音

调用【ELTE_SDK_VolUnMute(取消静音)】接口,实现在点对点呼叫、组呼、视频回传场景中取消静音,目前只支持对通话对端用户取消。

接口调用代码示例如下。

```
//cpp code
//构造静音参数xml字符串
CString strMuteParam;
strMuteParam.Append("<Content>");
strMuteParam. Append("<MuteParam>");
//呼叫类型: 0: 点呼 1: 组呼
                                   2: 视频回传
strMuteParam.Append("<CallType>");
strMuteParam. Append ("0");
strMuteParam. Append("</CallType>");
strMuteParam. Append("</MuteParam>");
strMuteParam. Append("</Content>");
//调度台对手持终端8895发起的语音点呼取消静音
ELTE_INT32 iRet = ELTE_SDK_VolUnMute("1001", strMuteParam);
//判断接口调用是否成功
if(0 == iRet)
    //接口调用成功
```

接口调用成功Ret返回为0,其他错误码可参考【eLTE产品错误码】。

发起环境侦听

调用【ELTE_SDK_StartEnvironmentListen(发起环境侦听)】调度台对终端用户周围环境进行侦听,终端的耳机自动打开,可以侦听到终端侧的声音。

□说明

环境侦听只能对终端用户发起。可以调用挂断语音点呼接口结束环境侦听。

接口调用示代码示例如下:

```
//cpp code
//调度台对手持终端8895发起环境侦听,只能对手持终端发起环境侦听
ELTE_INT32 iRet = ELTE_SDK_StartEnvironmentListen("8895");
//判断环境侦听接口调用是否成功
if(0 == iRet)
{
    //接口调用成功
}
```

接口调用成功Ret返回为0,其他错误码可参考【eLTE产品错误码】。

结束环境侦听

结束环境侦听调用【ELTE_SDK_P2PHangup(挂断语音点呼)】接口,参考挂断语音点呼。

发起缜密监听

调用【ELTE_SDK_StartDiscreetListen(发起缜密监听)】启动对调度台用户、终端用户或群组(包括普通组、派接组和动态组)的缜密监听,当该用户或群组有通话时,调度台用户能听到被缜密监听对象的通话内容,调度员只能听不能说。

```
//cpp code
//调度台4116对手持终端8890发起缜密监听,调度台发起缜密监听的对象可以是调度台用户ID、群组ID或手持终端
ID
ELTE_INT32 iRet = ELTE_SDK_StartDiscreetListen("8890");;
//判断接口调用是否成功
if(0 == iRet)
{
    //接口调用成功
}
```

接口调用成功Ret返回为0,其他错误码可参考【eLTE产品错误码】。

停止缜密监听

调用【ELTE_SDK_StopDiscreetListen(停止缜密监听)】停止对调度台用户、终端用户或群组(包括普通组和动态组)发起的缜密监听。

接口调用示代码示例如下:

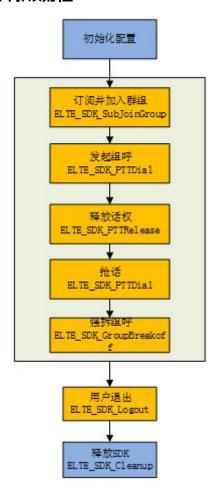
```
//cpp code
//调度台4116对手持终端8890停止缜密监听,调度台停止缜密监听的对象可以是调度台用户ID、群组ID或手持终端
ID
ELTE_INT32 iRet = ELTE_SDK_StopDiscreetListen("8890");
//判断接口调用是否成功
if(0 == iRet)
{
    //接口调用成功
}
```

接口调用成功Ret返回为0,其他错误码可参考【eLTE产品错误码】。

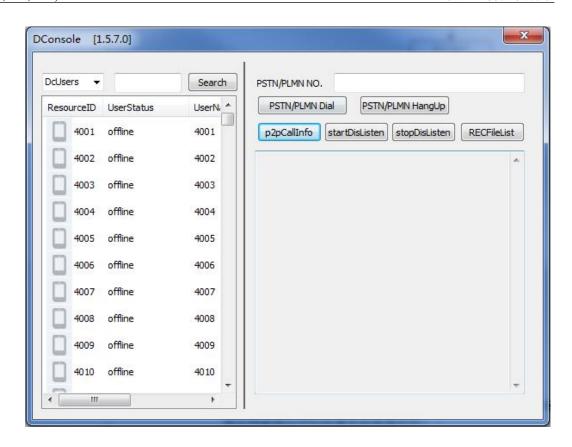
6.2.3 语音组呼场景

本章节的开发任务: 开发语音组呼的相关接口的方法和流程, 演示开发过程。

语音组呼接口调用流程



语音调度Sample界面:



前期条件

己完成初始化配置。

订阅并加入群组

调用【ELTE_SDK_SubJoinGroup(订阅并加入群组)】接口,订阅群组后,调度台可接收或发起群组组呼,未订阅群组则不能接收到群组组呼也不能发起该群组的组呼。

□说明

接口调用一次只能订阅一个群组。

接口调用示例:

接口调用成功后,解析回调函数【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】,回调函数返回的消息格式如下:

当StatusType=11, StatusValue=4003表示群组订阅成功。

取消订阅群组

调用【ELTE_SDK_UnSubscribeGroup(取消订阅群组)】接口,调度台取消订阅某个群组。

∭说明

调度台退订某个群组后,该群组将不能接受除订阅外的群组操作,除非调度台再次订阅该群组。 接口调用代码示例如下:

```
//cpp code

//取消订阅1001群组

ELTE_INT32 iRet = ELTE_SDK_UnSubscribeGroup("1001");

if(0 == iRet)

{

    //接口调用成功

}
```

接口调用成功后,解析回调函数中【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】,消息回调事件函数返回的消息格式如下。

当StatusType=11, StatusValue=4004表示群组取消订阅成功。

发起组呼或抢话

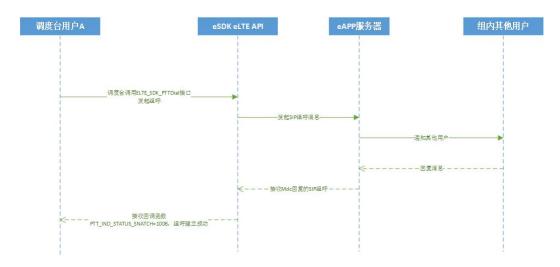
【ELTE_SDK_PTTDial(发起组呼或抢权)】接口用于发起组呼(固定、临时和动态组的组呼)或抢权,即调度台发起群组通话或在群组通话中抢权。若该群组当前已有活动通话,则调度台申请话权。



注意

调度台需要有抢话权限同时组内优先级要高于当前发言的用户才能抢话成功。

调度台发起组呼场景时序图如下:



接口调用代码示例如下:

```
//cpp code
//对1001群组发起组呼
ELTE_INT32 iRet = ELTE_SDK_PTTDial("1001");
if(0 == iRet)
{
    //接口调用成功
}
```

解析回调函数【ELTE_Event_NotifyGroupStatus(组呼状态变化事件通知)】,返回消息格式如下:

当GroupCallStatus=1006表示组呼建立成功,当GroupCallStatus=1011说明抢话成功,组呼其他状态值请参考【ELTE_Event_NotifyGroupStatus(组呼状态变化事件通知)】。

释放组呼话权

【ELTE_SDK_PTTRelease(释放话权)】接口用于组呼模式中发起组呼者或抢权成功者释放话语权。

接口调用成功后,解析回调函数【ELTE_Event_NotifyGroupStatus(组呼状态变化事件通知)】,返回消息格式如下:

收到GroupCallStatus=1012表示释放组呼话权成功,组呼其他状态值请参 【ELTE_Event_NotifyGroupStatus(组呼状态变化事件通知)】。

紧急组呼

【ELTE_SDK_PTTEmergency(发起紧急组呼)】调度台对已经存在的群组发起紧急呼叫。

接口调用代码示例如下:

```
//cpp code
//对群组1001发起紧急组呼
ELTE_INT32 strRet = ELTE_SDK_PTTEmergency ("1001");
//判断接口调用是否成功
if(0 == iRet)
{
    //接口调用成功
}
```

接口调用成功后,解析回调函数【ELTE_Event_NotifyGroupStatus(组呼状态变化事件通知)】,返回消息格式如下:

```
//xml code

<Content>

<GroupID>1001</GroupID>

<GroupCallStatus>1006</GroupCallStatus>

<Speaker>4116</Speaker>

<LocalPort>63710</LocalPort>

<RemotePort>28128</RemotePort>

<RemoteIP>120. 9. 22. 172</RemoteIP>

<SoundPtype>-1</SoundPtype>

<SpeakerName>4116</SpeakerName>

<CallID>0</CallID>

<SignalError>-1</SignalError>

<FromString></FomString>

<Content>
```

收到GroupCallStatus=1006表示紧急组呼成功,组呼其他状态值请参考【ELTE_Event_NotifyGroupStatus(组呼状态变化事件通知)】。

强拆组呼

【ELTE_SDK_GroupBreakoff(强拆组呼)】调度台强拆某个群组的组呼,该群组的当前活动呼叫会被强制结束。

```
//cpp code
//对群组1001发起强拆组呼
ELTE_INT32 strRet =ELTE_SDK_GroupBreakoff ("1001");
//判断接口调用是否成功
if(0 == iRet)
{
    //接口调用成功
}
```

接口调用成功后,解析回调函数【ELTE_Event_NotifyGroupStatus(组呼状态变化事件通知)】,返回消息格式如下:

收到GroupCallStatus=1008表示强拆组呼成功,组呼其他状态值请参考 【ELTE_Event_NotifyGroupStatus(组呼状态变化事件通知)】。

发起群组缜密监听

参考**发起缜密监听**,将入参变更为群组ID。

停止群组缜密监听

参考停止缜密监听,将入参变更为群组ID。

6.3 视频调度

6.3.1 概述

eSDK eLTE SDK开放视频调度接口,包括发起视频回传、接收视频回传、停止视频回传、发起视频分发、停止视频分发、获取视频墙ID列表、发起视频上墙、停止视频上墙等。

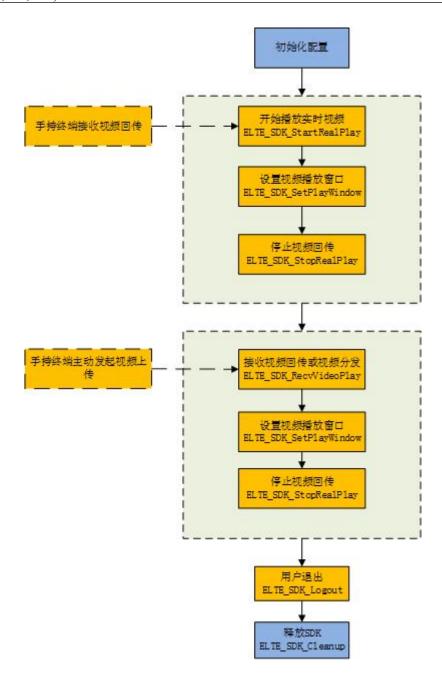
我们将按照下面的三种场景来分别介绍这些接口。

- 视频回传场景
- 视频分发场景
- 视频上墙场景

6.3.2 视频回传场景

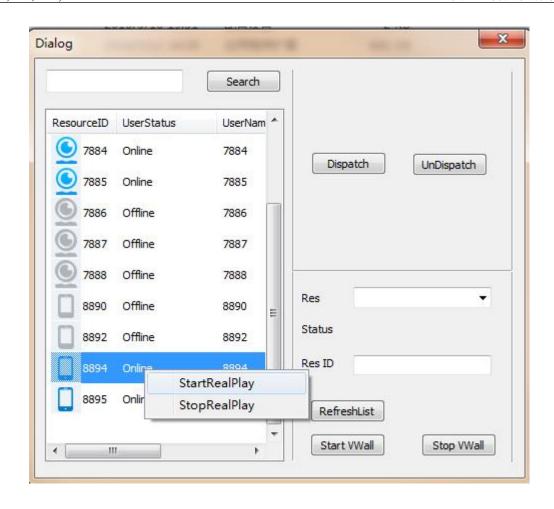
本章节将详细介绍视频回传场景下接口调用的流程方法和注意事项。

视频回传场景接口调用流程:



视频回传分为调度台主动向用户(手持终端或IVS摄像头)发起视频回传,和手持终端主动请求视频回传两种方式,调度台用户分别调用发起视频回传和接收视频回传两种接口来实现,两种方式只是视频回传的建立方式不同,整个流程的其他接口不变。

视频调度Sample界面:



前提条件

己完成初始化配置。

发起视频回传

调用【ELTE_SDK_StartRealPlay(开始播放实时视频)】接口向手持终端或IVS摄像头发起视频回传,调用接口前需要准备视频参数。

VideoFormat可以用CIF, QCIF, D1, 720P, 1080P五种视频格式,

CameraType表示摄像头类型,这个参数只对手持终端有效,0代表前置摄像头,1代表后置摄像头。

UserConfirmType表示是否需要用户确认视频回传,这个参数只有手持终端有效,0代表不需要用户确认,1代表需要用户确认

MuteType表示是否需要伴音,0代表需要伴音,1代表无伴音。

□说明

1.调度台用户需要有非确认方式视频回传权限,同时手持终端上需要启用允许自动确认视频回传 两个条件都具备时才能成功对手持终端进行无需用户确认的视频回传。

2.视频回传成功并不一定是按照入参视频格式回传,对于手持终端或ivs摄像头,eAPP会进行视频格式协商,在不支持的视频格式情况下,eAPP与手持终端和ivs摄像头默认按照实际支持的视频格式进行视频回传。详细需要解析回调函数iEventType=1, CallStatus=3003情况下

VideoFormatType参数。

以调度台向终端发起实时视频回传为例,信令时序图如下。



接口调用代码示例如下:

```
//cpp code
//构造一个视频回传入参xml字符串
CString strVideoParam;
strVideoParam.Append("<Content>");
strVideoParam.Append("<VideoParam>");
//视频格式: 0: V_CIF 1: V_QCIF 2: V_D1 3: V_720P 4: V_1080P
strVideoParam. Append ("<VideoFormat>");
strVideoParam. Append("4");
strVideoParam.Append("</VideoFormat>");
//摄像头类型:
              0: 前置摄像头 1: 后置摄像头
strVideoParam.Append("<CameraType>");
strVideoParam. Append("0");
strVideoParam.Append("</CameraType>");
//是否需要用户确认(通常情况下设置为0):
                                            0: 不需要用户确认 1: 需要用户确认
strVideoParam.Append("<UserConfirmType>");
strVideoParam.Append("0");
strVideoParam. Append("</UserConfirmType>");
//是否需要伴(通常情况下设置为0): 0:需要伴音
strVideoParam. Append("(MuteType>");
                                                   1: 无伴音
strVideoParam. Append ("0");
strVideoParam. Append("</MuteType>");
strVideoParam. Append("</VideoParam>");
strVideoParam.Append("</Content>");
//调用视频回传接口向8894终端发起视频回传
ELTE INT32 iRet = ELTE SDK StartRealPlay("8890", strVideoParam);
if(0 == iRet)
    //接口调用成功
```

接口调用成功后,实际视频回传发起是否成功需要解析回调函数 【ELTE_Event_NotifyP2pvideocallStatus(开始实况事件通知)】,接口调用返回xml消息格式如下:

```
//xml code

<Content>

<CallStatus>3003</CallStatus>

<Callee>8894</Callee>

<Caller>4116</Caller>

<LocalAudioPort>64586</LocalAudioPort>

<LocalVideoPort>64588</LocalVideoPort>

<RemoteAudioPort>28278</RemoteAudioPort>

<RemoteVideoPort>28282</RemoteVideoPort>

<RemoteIp>120.9.22.172</RemoteIp>

<Uri>8894</Uri>

<Channel>65535</Channel>

<SoundMute>0</SoundMute>

<UserConfirm>9</UserConfirm>
```

其中:

- CallStatus=3003表示对端已经接听呼叫,此时需要按照返回的参数信息执行媒体播放,在独立窗口中播放视频需要调用【ELTE_SDK_SetPlayWindow(设置播放实时视频窗口)】接口。
- CallStatus=3007表示对端拒绝接听。
- CallStatus=3008表示对端结束呼叫。
- 其他CallStatus值请参考【ELTE_Event_NotifyP2pvideocallStatus(开始实况事件通知)】。

设置视频播放窗口

调度台向手持终端或IVS摄像头发起视频回传成功后,需要调用 【ELTE_SDK_SetPlayWindow(设置播放实时视频窗口)】来设置视频播放窗口。

接口调用代码示例如下:

```
//cpp code
//窗口句柄,IDC_STATIC_RealPlay是播放窗口的ID
HWND hwnd = GetDlgItem(IDC_STATIC_RealPlay)->GetSafeHwnd();
//设置手持终端8890视频回传窗口
ELTE_INT32 iRet =ELTE_SDK_SetPlayWindow("8890", hwnd);
if(0 == iRet)
{
    //接口调用成功
}
```

停止视频回传

视频回传发起成功之后,调度台用户若要停止视频回传,需要调用【ELTE SDK StopRealPlay(停止播放实时视频)】接口。

接口调用代码示例如下。

```
//cpp code
//停止手持终端8890的视频回传
ELTE_INT32 iRet = ELTE_SDK_StopRealPlay("8890");
if(0 == iRet)
{
    //接口调用成功
}
```

接口调用成功后,实际视频是否结束需要解析回调函数

【ELTE_Event_NotifyP2pvideocallStatus(开始实况事件通知)】,当CallStatus=3009表示调度台主动停止视频回传成功。

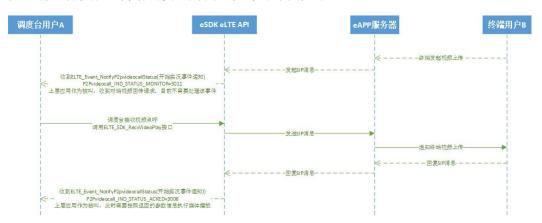
接收视频回传或视频分发

终端发起视频上传请求, SDK解析回调函数

【ELTE_Event_NotifyP2pvideocallStatus(开始实况事件通知)】, 收到3011= P2Pvideocall IND STATUS MONITOR事件通知。

- 如果第三方调度台要接收实时视频,则调用【ELTE_SDK_RecvVideoPlay(接收视频回传或视频分发)接口描述】接口接收视频回传或视频分发。
- 如果第三方调度台要拒接实时视频,则调用停止播放实时视频接口 【ELTE_SDK_StopRealPlay(停止播放实时视频)】。

以终端主动视频上传给调度台为例,信令时序图如下。



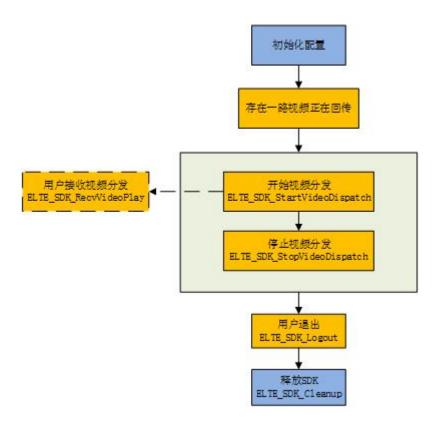
接收视频回传或视频分发接口调用示例:

```
//cpp code
//接收手持终端8890视频上传
ELTE_INT32 iRet =ELTE_SDK_RecvVideoPlay("8890");
if(0 == iRet)
{
    //接口调用成功
}
```

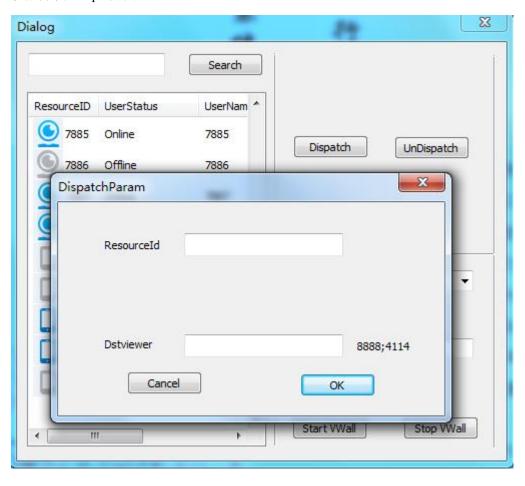
接口调用成功后,解析回调函数【ELTE_Event_NotifyP2pvideocallStatus(开始实况事件通知)】中CallStatus=3006表示接收视频回传或视频分发成功。其他CallStatus值请参考【ELTE Event NotifyP2pvideocallStatus(开始实况事件通知)】。

6.3.3 视频分发场景

本章节详细介绍视频分发场景中接口调用的方法、注意事项和调用流程。 视频分发场景中接口调用流程:



视频分发Sample界面:



前提条件

- 1. 已完成初始化配置。
- 2. 调度台已经存在一路正在播放的实时视频。

开始视频分发

调度台调用【ELTE_SDK_StartVideoDispatch(开始视频分发)】接口可以将一路正在回传的视频分发给其他的终端或调度台。当调度台用户挂断该路回传的视频时,分发也自动结束。

接口调用入参包括分发的视频格式、视频源ID,目的用户ID等。



注意

1.该接口调用前确保有一路视频正在回传

2.视频分发的视频格式与视频回传的视频格式保持一致。视频分发中Fmtvalue参数取值,由消息回调函数上报14.2 ELTE_Event_NotifyP2pvideocallStatus (开始实况事件通知)中事件类型CallStatus=3003的事件消息体中VideoFormatType的参数值决定的。

3.视频分发的目的用户可以是多个,视频源用户只能一个。

接口调用代码示例如下:

```
//cpp code
//构造视频分发入参XML
CString strDispatchParam;
strDispatchParam.Append("<Content>");
//视频格式,取值只能为"NO"或"CIF",其中,NO为原码转发,CIF为转码分发
strDispatchParam.Append("<Fmtvalue>");
strDispatchParam.Append("NO");
strDispatchParam.Append("</Fmtvalue>");
//视频源用户ID
strDispatchParam.Append("<DispatchNum>");
strDispatchParam.Append("8894");
strDispatchParam.Append("</DispatchNum>");
//视频分发目的用户列表
strDispatchParam.Append("<Dstviewerlist>");
strDispatchParam.Append("<Dstviewer>");
strDispatchParam.Append("4135");
strDispatchParam.Append("</Dstviewer>");
strDispatchParam.Append("</Dstviewerlist>");
//保留参数
strDispatchParam. Append("<Channel>");
strDispatchParam. Append("</Channel>");
strDispatchParam.Append("</Content>");
//调用视频分发接口
ELTE_INT32 iRet = ELTE_SDK_StartVideoDispatch("8894", strDispatchParam);
if(0 == iRet)
    //接口调用成功
```

接口调用成功后,视频分发的目的用户是否接受通过解析回调函数 【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】StatusType和StatusValue来确定,例如:

● 资源状态变化事件 ResourceID=8892, StatusType=22, StatusValue=4021表示调度 台分发给8892后,8892正在振铃音。

- ResourceID=8892, StatusType=22, StatusValue=4022表示调度台分发给8892后, 8892终端已经接受视频分发。
- ResourceID=8892, StatusType=22, StatusValue=4023表示调度台分发给8892后, 8892终端已经拒绝视频分发或视频分发未接听超时。

其他状态值请参考【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】。

停止视频分发

当调度台用户发起对一个或多个终端用户的视频分发后,需要终止其中某一路视频分发,则调用【ELTE SDK StopVideoDispatch(停止视频分发)】接口,即单点挂断。

接口入参包括发起视频分发的调度台ID、视频源ID、需要挂断的视频分发用户ID。

接口调用代码示例如下:

```
//cpp code
//构造视频分发停止的入参xml
CString strDispatchParam;
strDispatchParam. Append("<Content>");
//视频源ID
strDispatchParam.Append("<ResourceId>");
strDispatchParam. Append ("8894");
strDispatchParam.Append("</ResourceId>");
//带停止的视频分发目的用户ID
strDispatchParam.Append("<UserId>");
strDispatchParam. Append("4135");
strDispatchParam. Append("</UserId>");
strDispatchParam. Append("</Content>");
//调用停止视频分发接口,第一个入参是视频源ID
//调用停止视频分发接口
ELTE_INT32 iRet = ELTE_SDK_ StopVideoDispatch ("8894", strDispatchParam);
if(0 == iRet)
    //接口调用成功
```

接口调用成功后,停止视频分发是否成功通过解析回调函数

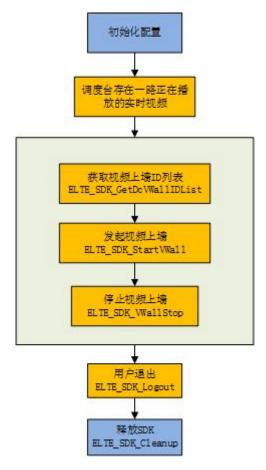
【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】中的StatusType和StatusValue来确定,例如:

ResourceID=8892, StatusType=22, StatusValue=4023表示调度台结束给8892的视频分发成功,其他状态值请参【ELTE_Event_NotifyResourceStatus(群组关系状态变化事件通知)】。

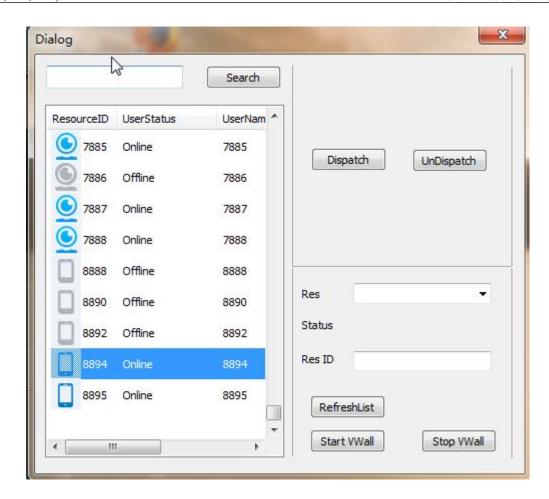
6.3.4 视频上墙场景

本章节主要介绍视频上墙的前提条件,视频上墙业务接口调用流程和注意事项。

视频上墙业务接口调用流程:



视频上墙Sample界面:



前提条件

- 1. 己完成初始化配置。
- 2. 调度台已经存在一路正在播放的实时视频。
- 3. 调度机上已经成功添加解码器,并且完成调度员与解码器关联配置。

获取视频上墙 ID 列表

视频上墙前需要先调用【ELTE_SDK_GetDcVWallIDList(获取视频墙ID列表)】获取视频上墙的ID列表以及视频通道的状态。



注意

若使用完该接口,需要调用【ELTE_SDK_ReleaseBuffer(释放内存)】释放内存接口将其资源释放掉。

接口调用代码示例如下:

```
//cpp code
//定义一个字符串指针作为接口函数入参
ELTE_CHAR *pStr = NULL;
//调用获取视频上墙ID列表
ELTE_INT32 iRet = ELTE_SDK_GetDcVWallIDList(&pStr);
if(0 == iRet)
```

```
{
//成功后释放内存
ELTE_SDK_ReleaseBuffer(pStr);
}
```

接口调用成功后,函数中pStr带回一个XML字符串,格式如下:

```
//xml code
//查询视频墙列表和视频墙的状态,视频墙ID状态: 1: 初始化 4022: 已经占用 4023: 空闲
<Content>
    <VWallIDList>
       <VWallID>
           <Dst0b iId>99910001/Dst0b iId>
           <IDState>1</IDState>
           <Alias>IVS1</Alias>
       </WwallID>
       <VWallID>
           <Dst0bjId>99910002</pst0bjId>
           <IDState>1</IDState>
           <Alias>IVS2</Alias>
       </WallID>
       <VWal1ID>
           <Dst0bjId>99910003</pst0bjId>
           <IDState>1</IDState>
           <Alias>IVS3</Alias>
       </WallID>
       <VWallID>
           <Dst0bjId>99910004</pst0bjId>
           <IDState>1</IDState>
           <Alias>IVS4</Alias>
       </WallID>
    </WallIDList>
</Content>
```

您可通过视频通道ID的状态判断发起视频上墙或停止视频上墙是否成功

发起视频上墙

调用【ELTE_SDK_VWallStart(发起视频上墙)】接口发起视频上墙,入参包括视频墙通道号和待上墙的视频源ID。



注音

在发起视频上墙前必须要先调用查询视频上墙列表接口,并选择空闲的通道号。

接口调用示例:

```
//cpp code
//构造视频上墙参数
CString strVideoChannelStart;
strVideoChannelStart. Append("<Content>");
strVideoChannelStart. Append("<VideoParam>");
//视频墙通道ID
strVideoChannelStart. Append("<Dst0bjId>");
strVideoChannelStart. Append("99910001");
strVideoChannelStart. Append("</Dst0bjId>");
//保留参数
strVideoChannelStart. Append("<<Tent>");
strVideoChannelStart. Append("<<*tent>");
strVideoChannelStart. Append("<</tent>");
strVideoChannelStart. Append("</tent>");
strVideoChannelStart. Append("</tent>");
strVideoChannelStart. Append("</tent>");
strVideoChannelStart. Append("</tent>");
strVideoChannelStart. Append("</tent>");
//调用视频上墙接口,第一个参数是视频源ID
ELTE_INT32 iRet = ELTE_SDK_VWallStart("8890", strVideoChannelStart);
```

接口调用成功后通常可通过查看视频墙通道的状态来判断视频上墙最终是否成功,如果视频墙通道的状态是4022,说明发起视频上墙成功

停止视频上墙

调用【ELTE_SDK_VWallStop(终止视频上墙)】停止视频视频上墙,入参包括视频墙通道号和待停止上墙的视频源ID。

接口调用代码示例如下:

```
//cpp code
//构造待终止的视频源和视频墙参数
CString strVideoChannelStop;
strVideoChannelStop. Append("〈Content〉");
//视频墙通道ID
strVideoChannelStop. Append("〈DstObjId〉");
strVideoChannelStop. Append("99910001");
strVideoChannelStop. Append("〈/DstObjId〉");
strVideoChannelStop. Append("〈/Content〉");
//调用终止视频上墙接口,第一个参数是视频源ID
ELTE_INT32 iRet = ELTE_SDK_VWallStop ("8890", strVideoChannelStop);
if(0 == iRet)
{
//接口调用成功
}
```

接口调用成功后通常可通过查看视频墙通道的状态来判断终止视频上墙最终是否成功,如果视频墙通道的状态是4023,说明终止视频上墙成功。

6.4 短数据

6.4.1 概述

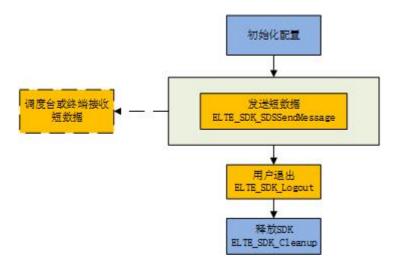
华为 eSDK eLTE SDK开放eLTE宽带集群系统的短数据业务接口,提供短数据的发送和接收接口,短数据发送【ELTE_SDK_SDSSendMessage(发送短数据)】使用接口,短数据的接收通过解析回调函数中【ELTE_Event_NotifySDSReport(短信/彩信接收上报事件通知)】来接收短数据。

短数据分为普通短信、彩信、状态短信三种,发送短数据仅支持普通短信和彩信,状态短信一般都是终端向调度台发送状态短信。

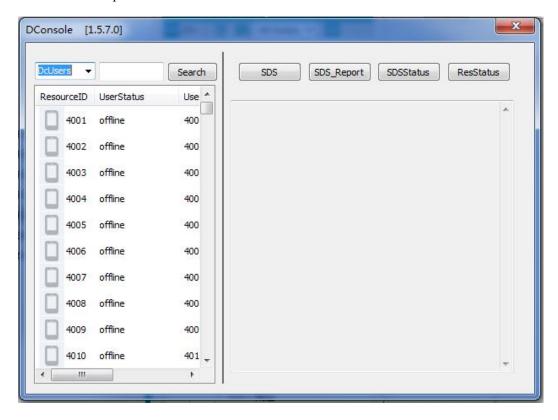
6.4.2 发送短数据场景

本章节介绍发送短数据接口的调用方法和流程,以及注意事项。

发送短数据接口调用流程:



发送短彩信Sample界面:



前提条件

已经完成初始化配置。

发送短数据

调用【ELTE_SDK_SDSSendMessage(发送短数据)】接口可以实现向其他调度台、终端以及群组发送普通短信和彩信的功能,接口入参包括发件人ID、短数据内容、收件人ID、附件地址、MsgID(保留参数,默认不写)。下面来分别介绍发送短信和发送彩信的区别。

● 发送普通短信

短数据类型MsgBody=0001表示普通短信,MsgBody和Receiver是必填项,Receiver可以是多个接受者,接收者可以是调度台用户、终端用户、群组用户等,AttachFileList不填写,MsgID默认不填写。

接口调用代码示例如下:

```
//cpp code
//构造接口入参xml字符串
CString strMsg;
strMsg. Append("<Content>");
//短数据类型: 0001,表示可以点对点或发群组短信;
strMsg.Append("<SDSType>");
strMsg. Append ("0001");
strMsg.Append("</SDSType>");
//发送短数据的内容
strMsg.Append("<MsgBody>");
strMsg.Append("Test Send Message.");
strMsg. Append("</MsgBody>");
//收件人,可以是用户ID或群组ID,多个ID用英文分号分隔,例如1001;1002;1003
strMsg.Append("<Receiver>");
strMsg. Append ("8895");
strMsg. Append("</Receiver>");
//附件地址列表,目前只支持一个附件。当SDSType=0001,则AttachFileList节点不存在。
strMsg.Append("<AttachFileList>");
strMsg.Append("<AttachFile>");
strMsg.Append("</AttachFile>");
strMsg. Append("</AttachFileList>");
//指定短消息ID, 可以默认不填
strMsg. Append("<MsgId>");
strMsg. Append("</MsgId>");
strMsg.Append("</Content>");
//调用发送短数据接口,第一个参数是当前登录的调度台ID
ELTE_INT32 iRet =ELTE_SDK_SDSSendMessage("4120", strMsg);
if(0 == iRet)
    //接口调用成功
```

● 发送彩信

短数据类型MsgBody=0004表示彩信,MsgBody、Receiver、AttachFile是必填项,AttachFile只能有且只有一个,Receiver可以是多个接受者,接收者可以是调度台用户、终端用户、群组用户等,MsgID默认不填写。

代码示例:

```
//cpp code
//构造接口入参xml字符串
CString strMsg;
strMsg.Append("<Content>");
//短数据类型为0004,表示可以点对点或发群组短信或者彩信;
strMsg. Append("<SDSType>");
strMsg. Append("0004");
strMsg.Append("</SDSType>");
//发送短数据的内容,SDSType=0004,则MsgBody可选,不管是短信还是彩信内容都可以发送空的;MsgBody短信
内容最大支持输入1000个字节
strMsg.Append("<MsgBody>");
strMsg.Append("Test Send Message.");
strMsg.Append("</MsgBody>");
//收件人,可以是用户ID或群组ID,多个ID用英文分号分隔,例如1001;1002;1003
strMsg. Append("<Receiver>");
strMsg. Append("8895");
strMsg.Append("</Receiver>");
//附件地址列表,目前只支持一个附件,当SDSType=0004,则AttachFileList节点必选,支持文本文件、图片、
视频片段、压缩包等,附件上限为2M。
strMsg.Append("<AttachFileList>");
strMsg.Append("<AttachFile>");
strMsg. Append("D:\picture.jpg");
```

```
strMsg. Append("</AttachFile>");
strMsg. Append("</AttachFileList>");
//指定短消息ID,可以默认不填
strMsg. Append("<MsgId>");
strMsg. Append("</MsgId>");
strMsg. Append("</Content>");
//调用发送短数据接口,第一个参数是当前登录的调度台ID
ELTE_INT32 iRet =ELTE_SDK_SDSSendMessage("4120", strMsg);
if(0 == iRet)
{
    //接口调用成功
}
```

短信发送成功后接口调用成功返回0,对端是否接收到了短信呢,我们需要解析回调函数【ELTE_SDK_SetEventCallBack(设置消息回调函数)】,得到一个xml消息格式如下:

如果SdsReceiver=接受者ID,SdsRetCode=0,表示发送成功,SdsRetCode=0xff表示接收端没有接收,其他的状态请参考【EVENT_NOTIFY_SDS_SEND_STATUS(短数据发送状态事件通知)】。



鐅生

为了保证用户信息安全,在进行日志记录时,发送和接收的短数据内容都不要记录在日志中。

接收短数据

其他用户向调度台发送短数据,调度台用户要怎么接收呢?

首先解析回调函数【ELTE_Event_NotifySDSReport(短信/彩信接收上报事件通知)】,得到一个xml消息格式如下。

短数据接收可以接收普通短信息、彩信和状态短信三种类型,代码调试时可分别验证 这三种情况。



警告

为了保证用户信息安全,在进行日志记录时,发送和接收的短数据内容都不要记录在 日志中。

6.5 订阅终端 GIS

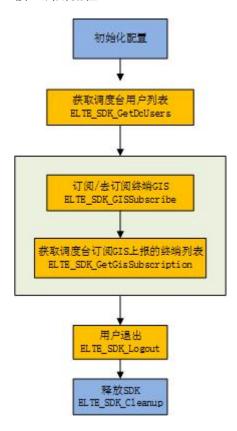
6.5.1 概述

华为 eSDK eLTE SDK开放eLTE宽带集群系统的终端GIS订阅业务接口,调度台订阅手持终端GIS后,终端GPS定位后将位置信息周期上报给调度台。调度台通过解析回调函数【ELTE_Event_NotifyGISStatus(GIS状态事件通知)】来查看调度台订阅终端的状态,通过解析【ELTE_Event_NotifyGISReport(GIS信息事件通知)】来获取终端的位置信息。

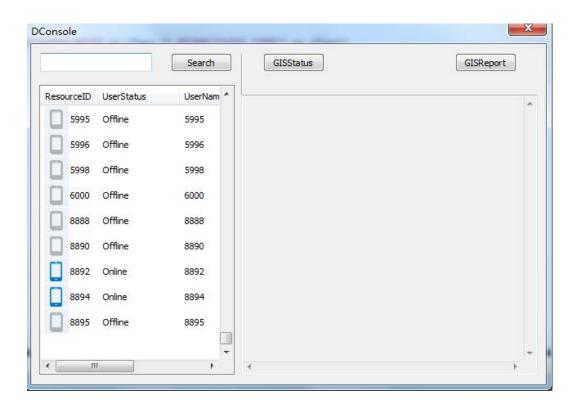
6.5.2 订阅终端 GIS 位置信息

本章节我们来介绍调度台订阅终端GIS接口调用方法和流程以及注意事项。

接口调用流程:



订阅终端GIS位置Sample界面:



前提条件

- 1. 已经完成初始化配置。
- 2. 手持终端"设置-定位服务"中允许访问我的位置信息。

获取调度台用户列表

参考【**获取调度台用户列表**】。其中用户类型UserCategory=9是PTT用户,只能对PTT用户发起GIS订阅和去订阅。

订阅/去订阅终端 GIS 场景

调度台需要访问手持终端的位置信息,需要调用【ELTE_SDK_GISSubscribe(订阅/去订阅GIS终端)】接口,入参有三个,分别是调度台ID,订阅类型,订阅终端列表。订阅类型SubType=7表示订阅,SubType=8代表去订阅,订阅终端列表可以是多个终端,之间用分号分割,最多200个终端。

订阅8890、8894、8892三个终端GIS的代码示例如下。

```
//cpp code
//构造GIS订阅入参xml字符串
CString strGISParam;
strGISParam. Append("〈Content〉");
strGISParam. Append("〈GISParam〉");
//订阅类型: 7: 代表订阅 8: 代表不订阅
strGISParam. Append("〈SubType〉");
strGISParam. Append("不");
strGISParam. Append("〈SubType〉");
//资源列表,多个资源ID用分号分隔,最多200个。(例如1001;1002;1003)
strGISParam. Append("〈ResourceList〉");
strGISParam. Append("《ResourceList〉");
strGISParam. Append("《ResourceList〉");
strGISParam. Append("〈ResourceList〉");
//预留参数,可不填写
strGISParam. Append("〈Subscriber〉");
```

```
strGISParam. Append("");
strGISParam. Append("</subscriber>");
strGISParam. Append("</GISParam>");
strGISParam. Append("</content>");
//调用订阅终端GIS订阅接口
ELTE_INT32 iRet = ELTE_SDK_GISSubscribe("0", strGISParam);
if(0 == iRet)
{
    //接口调用成功
}
```

调度机eMDC是否接受调用请求需要检查回调函数中的事件类型iEventType =9,通过校验ResourceID=订阅调度台ID,AckStatus=终端ID:0来确定终端回复给调度机是否接收到GIS订阅命令。解析消息回调事件函数中【ELTE_Event_NotifyGISStatus(GIS状态事件通知)】得到xml消息格式如下,说明eMDC处理订阅终端GIS请求成功并已经通知到订阅的终端。

终端在接收到GIS订阅命令后,进行GPS定位,将位置信息周期上报或在紧急情况下发送调度机,调度机处理后将位置信息上报到调度台。此时调度台需要解析消息回调事件函数事件类型iEventType =8【ELTE_Event_NotifyGISReport(GIS信息事件通知)】得到xml消息格式如下。

□ 说明

GIS订阅成功后,调度机默认将终端上次的位置信息上报到调度台,待终端GPS定位位置成功后按照周期上报或特殊事件上报。所以调度台第二次收到的终端位置信息才是终端实际的位置。以EP820终端为例,终端通知栏显示如下说明终端GPS定位成功,按周期上报。



获取调度台订阅 GIS 上报的终端列表

调用【ELTE_SDK_GetGisSubscription(获取订阅GIS的终端列表)】接口,获取本调度台已订阅GIS的终端列表。接口调用代码示例如下。

```
//cpp code
//定义一个指针用于返回查询结果
ELTE_CHAR* pQueryResult = NULL;
//调用获取本调度台己订阅GIS的终端列表
ELTE_INT32 iRet = ELTE_SDK_GetGisSubscription("4116", &pQueryResult);
if(0 == iRet)
{
//接口调用成功后,释放内存空间
ELTE_SDK_ReleaseBuffer(pQueryResult);
}
```

pQueryResult返回值xml消息格式如下:

```
//xml code
<Content>
         <GisQuerySubList>
         <GisQuerySubscription>
             <UeID>8890</UeID>
             <UserName>8890</UserName>
         </GisQuerySubscription>
         \langle {\tt GisQuerySubscription} \rangle
             <UeID>8894</UeID>
             <UserName>8894
         </GisQuerySubscription>
         <GisQuerySubscription>
             \langle UeID \rangle 8895 \langle /UeID \rangle
             <UserName>8895</UserName>
         </GisQuerySubscription>
    </GisQuerySubList>
</Content>
```

7 定位指南

错误码获取方法

1. 接口返回

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。如果成功,则返回0。其他表示失败,该返回值即为错误码。

2. 日志获取

- 日志路径:

eSDK eLTE的接口返回值可以通过查看接口日志文件eSDK-eLTE.interface.log获取。

获取方法:

接口日志文件里记录了调用的每个接口调用的时间、日志级别、接口入参以及调用结果。

以登录eLTE_SDK_Login接口为例。

2016-05-28 16:31:18 399| INFO|eSDK-eLTE-SDK-Windows|1|Native| ELTE_SDK_Login||||2016-05-28 16:31:17 930|2016-05-28 16:31:18 398|UserID:4114, ServerIP:172.22.9.105, LocalIP:172.24.4.212, ServerSIPPort:5060|0|

∭说明

- 1. eLTE SDK Login: 搜索接口关键字 "eLTE SDK Login"可以查看该接口调用信息。
- 2. 最后一列0:接口返回值。

错误信息查询方法

参考接口文档eSDK eLTE API 接口参考(Windows, C++),列出了所有错误码信息。 根据接口返回的错误码,查询相对应的错误描述。

日志分析

Debug日志开关

● C/S模式:

设置资源包路径下的eSDKClientLogCfg.ini文件中的LogLevel_Run的值,为0则会打印debug日志信息。设置完毕后,重启程序。

获取Debug日志

● C/S模式:

加载资源包所在路径下的logs\

说明:日志路径是SDK的默认路径,若有需要,可以调用设置日志文件路径 eLTE_SDK_SetLogPath接口自定义日志路径

logs文件夹下共包括eSDK-eLTE.interface.log、eSDK-eLTE.operation.log、eSDK-eLTE.run.log这三个文件,如下图所示。



eSDK-eLTE.interface.log:接口调用出参、入参以及返回值信息

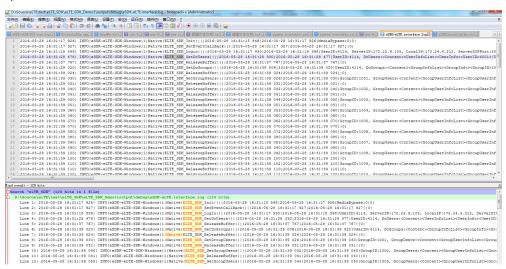
eSDK-eLTE.operation.log:接口调用操作信息,一般很少涉及

eSDK-eLTE.run.log:接口调用运行信息,较详细

开启debug日志后,eSDK-eLTE.run.log这个文件里的日志信息最丰富,是定位问题最关键的日志。

日志解析

1. 全局搜索关键字 "eLTE_SDK_",搜索结果即为全部调用的SDK接口,如下图所示。



2. 日志分析:

2016-06-16 11:00:27 657|ERROR|eSDK-eLTE-SDK-Windows|1|Native|ELTE_SDK_Login||||2016-06-16 11:00:27 608|2016-06-16 11:00:27 657|UserID:4114, ServerIP:172.22.9.105, LocalIP:172.24.4.249, ServerSIPPort:5060|-40134|

- 2016-06-16 11:00:27 657: 日志打印时间
- ERROR: 日志级别
- eSDK-eLTE-SDK-Windows: 模块名
- 1:接口类型

- Native: 协议类型
- ELTE SDK Login: 接口名
- 2016-06-16 11:00:27 608:接口调用时间
- 2016-06-16 11:00:27 657:接口返回时间
- UserID:4114, ServerIP:172.22.9.105, LocalIP:172.24.4.249, ServerSIPPort:5060:接口入参
- -40134: 接口返回值

本例是eLTE_SDK_Login登录接口调用出错的场景,当程序出错时会马上返回出来,所以我们可以根据这一特点来分析出错的具体原因。

- 1. 在Interface目志中找到"eLTE_SDK_Login"。
- 2. 检查调用eLTE_SDK_Login接口代码发现输入的本地IP有误。重新输入正确IP后登录成功。

8 开发指南修订记录

修订记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

日期	修订版本	描述
2016-10-25	V2.1.00	V200R001C00版本文档发 布。 新增 典型业务开发场景 部 分接口调用时序图。
2016-06-15	V1.5.70	修改内容包括: 1. eSDK eLTE OCX是什么 2. 开发指南有什么 3. 增加相关资源和下载链接 4. 增加helloworld 5. 细化典型业务开发场景 6. 增加定位指南
2015-11-09	V1.5.50	V100R005C50版本文档发 布。
2015-08-27	V1.5.30	V100R005C30版本文档发 布。