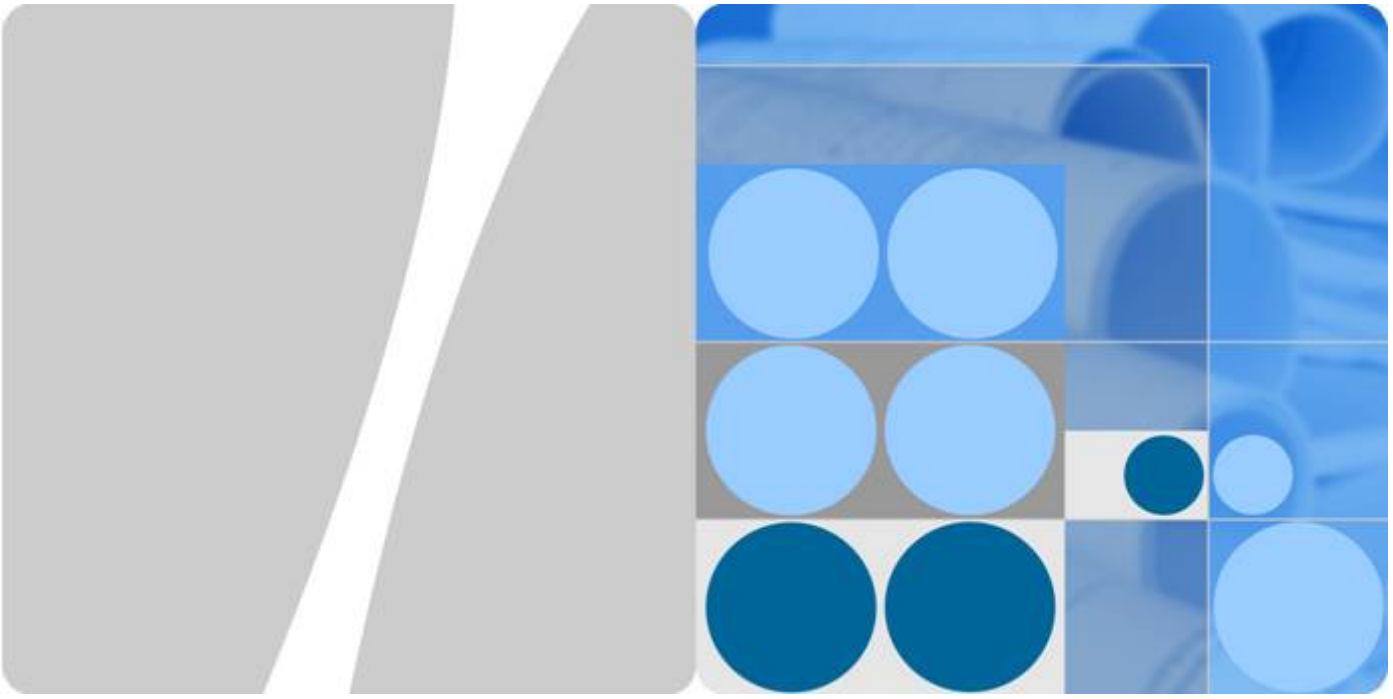


eSDK eLTE

V200R001C00

开发指南 01(PC,OCX,C++)

文档版本　01

发布日期　2016-10-25

华为技术有限公司

HUAWEI

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： 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 OCX 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 OCX 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，建议参考本开发指南。

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，建议使用**eSDK eLTE API 开发指南(Windows，C++)**。

eSDK eLTE OCX 包含什么

eSDK eLTE OCX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。主要包括以下内容：

- **OCX包**

eSDK eLTE二次开发使用的OCX SDK包，提供OCX安装文件和接口参考文档。详细请参见**SDK下载路径**。

- **Sample Codes**

  华为SDK提供一系列Sample codes演示如何调用接口，完成视频监控业务开发。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 OCX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OCX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，并帮助您快速开发一个基于eSDK eLTE OCX接口的小程序，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OCX版本号、加载OCX控件、设置消息消息回调事件函数、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**：介绍eSDK eLTE OCX控件开放的典型业务开发场景，详细介绍开发流程、代码示例以及注意事项等信息。
5. **定位指南**：开发过程中常见问题的定位方法。
6. **开发指南修订记录**：各版本开发指南更新细节。

阅读建议

- 如果您刚开始了解和使用eSDK eLTE OCX，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**典型业务开发场景**。
- 如果本指南中没有您需要的接口功能，请参考**eLTE OCX 接口参考(Windows，C++)**。
- 如果您在使用eSDK eLTE OCX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**技术支持**。

  说明

  本开发指南配套的OCX版本号1.5.70，如果需要查看其它版本OCX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 OCX开发指南**”，选择对应版本进行下载。

3 相关资源

3.1 华为开发者社区

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 OCX业务包**”，选择对应版本进行下载。

3.3 Sample Codes

代码样例包括5种**典型业务开发场景**的Sample Codes，演示如何调用OCX接口实现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 OCX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 OCX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_Video	基于eLTE OCX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 OCX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_GIS	基于eLTE OCX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。

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 OCX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

您可以进入华为开发者社区资源中心，点击“**开发文档** > **接口文档** > **宽带集群** > **eAPP610** > **eSDK eLTE OCX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**。

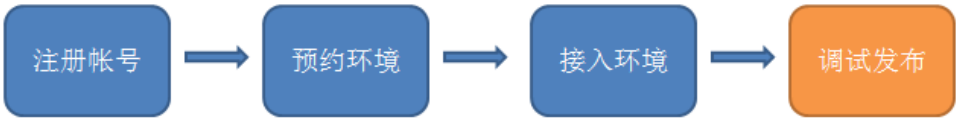
3.6 免费申请远程实验室

为什么要申请远程实验室

为了便于开发者体验华为网络和在华为设备上进行二次开发，远程实验室提供了多种产品网络环境，并通过体验demo供用户体验和调试编译程序，大大降低了用户二次开发的成本。

申请远程实验室环境

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。

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 8828 000
  b. 华为技术支持邮箱：**esdk@huawei.com**

4 Hello World

Hello World 开发流程

本示例以C++语言进行eSDK eLTE OCX 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 准备资源

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_OCX_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 OCX业务包**”，选择对应版本进行下载。

说明

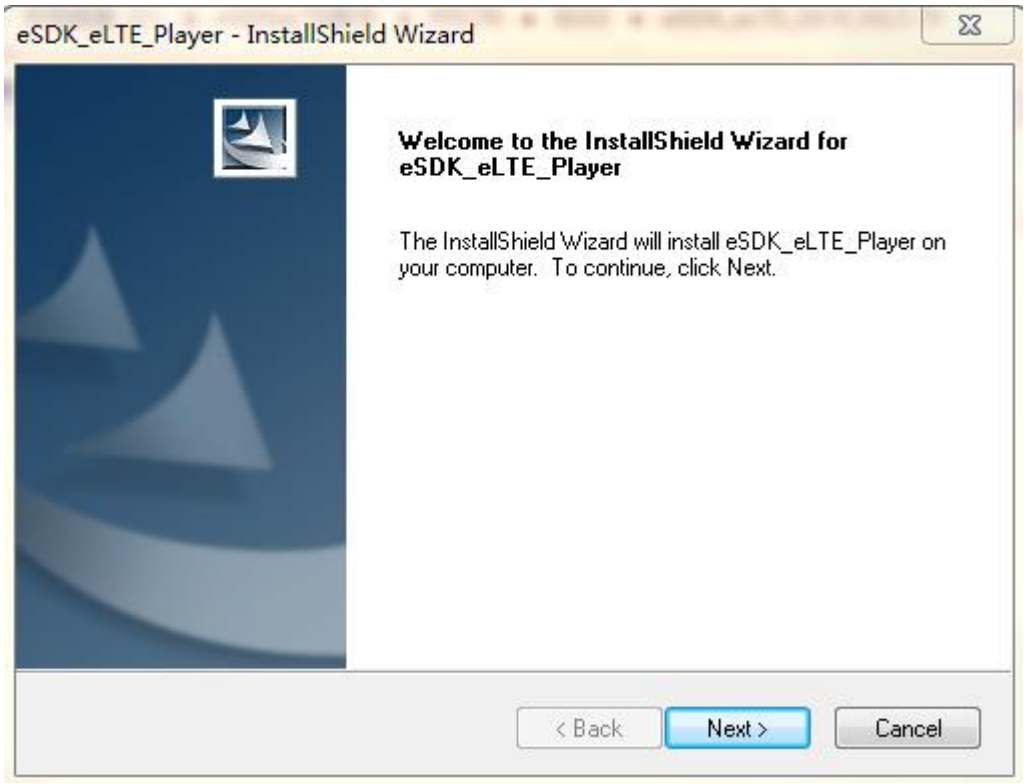
V2.1.00表示华为eSDK eLTE版本号，如：V2.1.00表示版本号为V200R001C00。

eLTE 环境

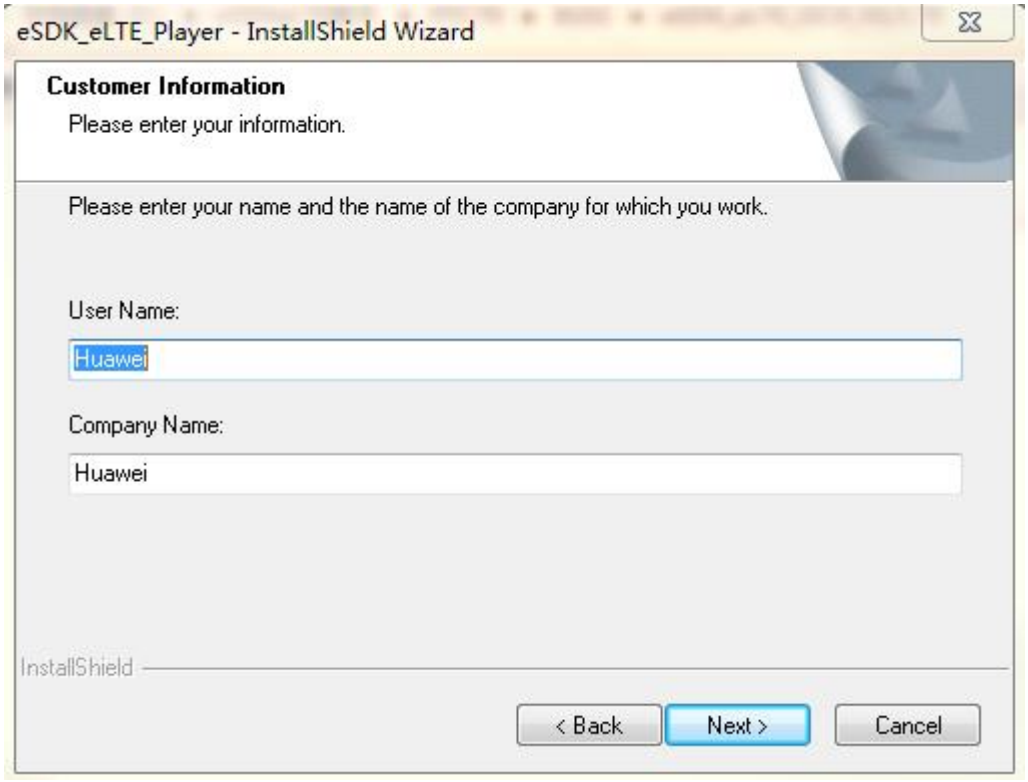
若您还未购买华为eLTE宽带集群产品，可登录华为远程实验室，**免费申请远程实验室环境**，体验eLTE宽带集群和调试二次开发代码。

4.2 注册 OCX 控件

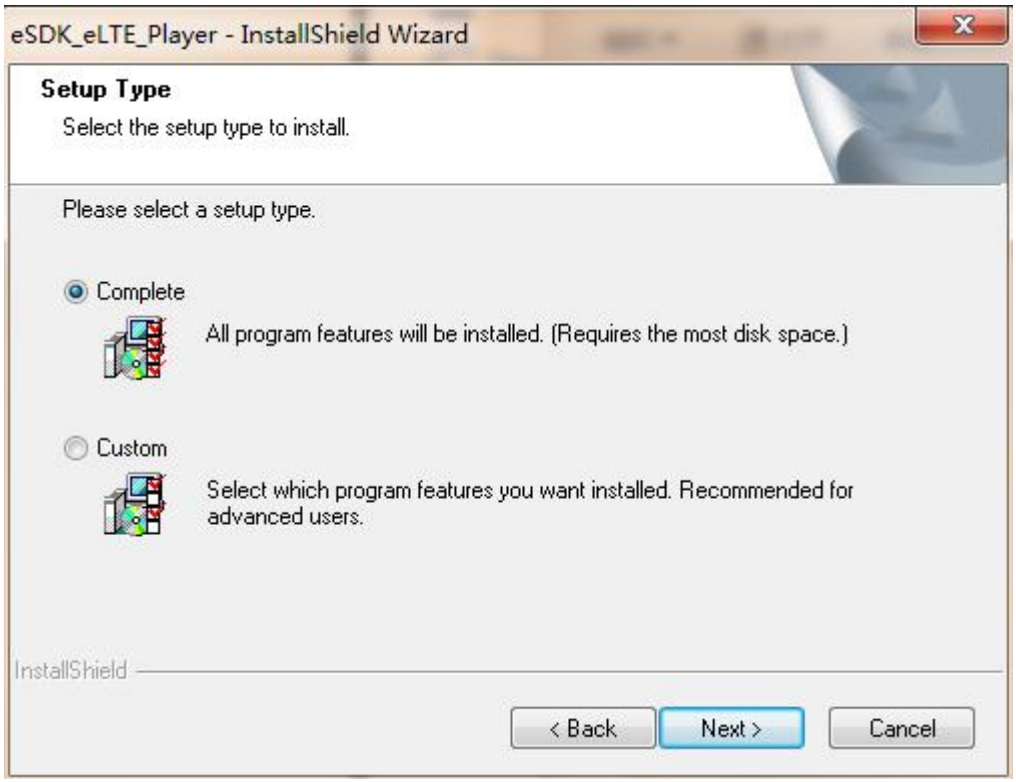
步骤1 将获取到的SDK业务包“eSDK_eLTE_OCX_V2.1.00.zip”解压并以管理员的身份运行.exe文件，系统弹出如下图所示提示信息，单击“**Next**”。



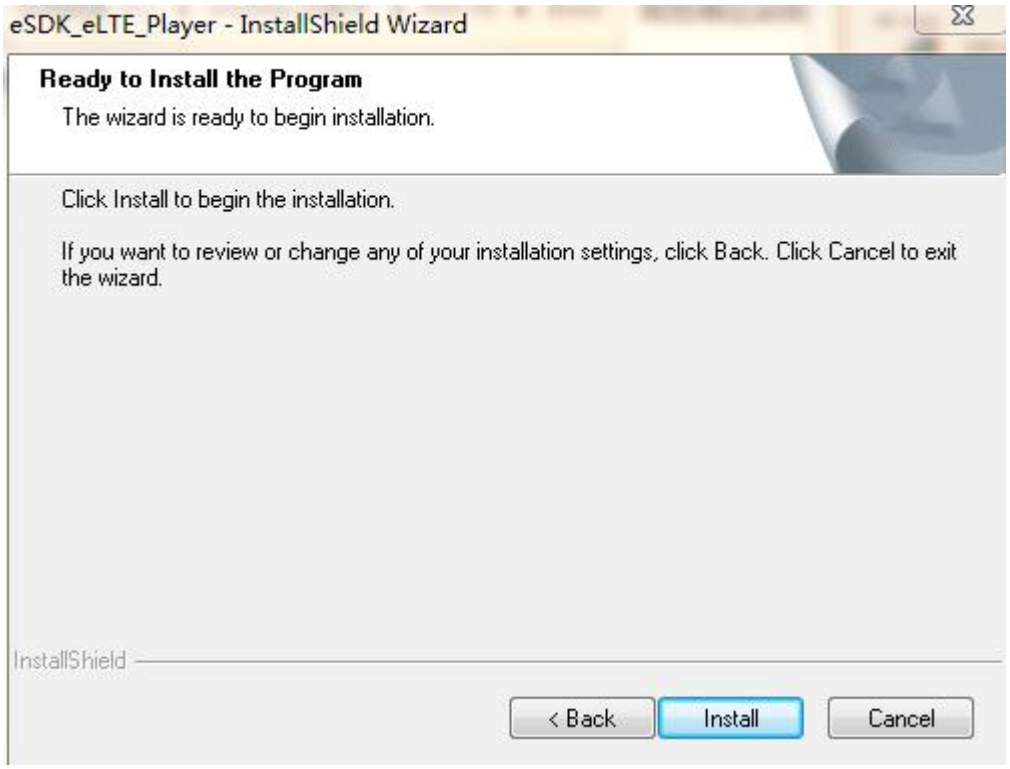
步骤2 输入用户名和公司名，单击“**Next**”，如下图所示。



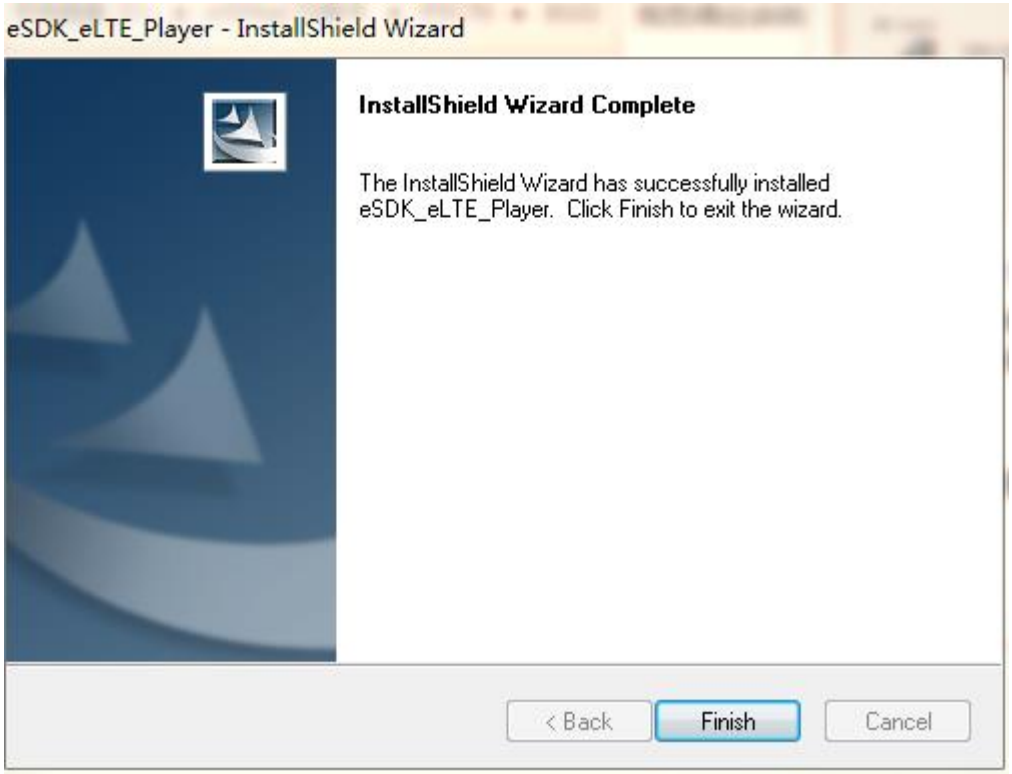
步骤3 选择安装类型“**Complete**”则默认安装在C:/eSDK_eLTE路径下，选择安装类型“**Custom**”则可修改安装路径，我们选择完全安装，如下图所示。



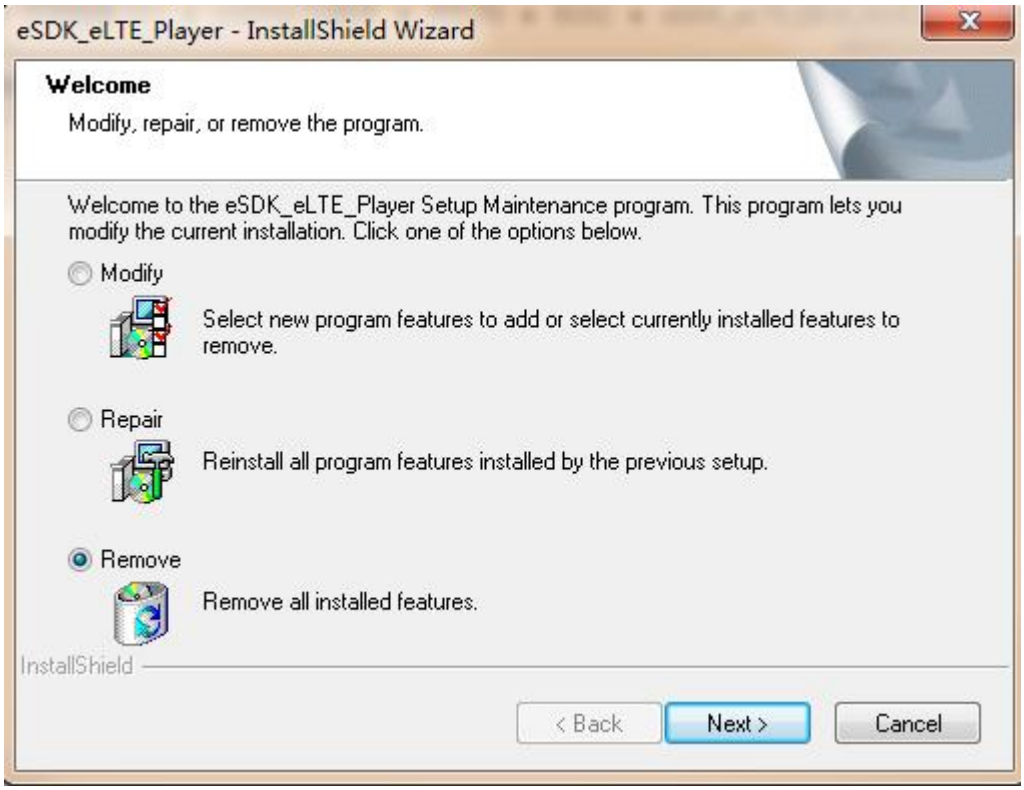
步骤4 单击“**Install**”开始安装，如下图所示。



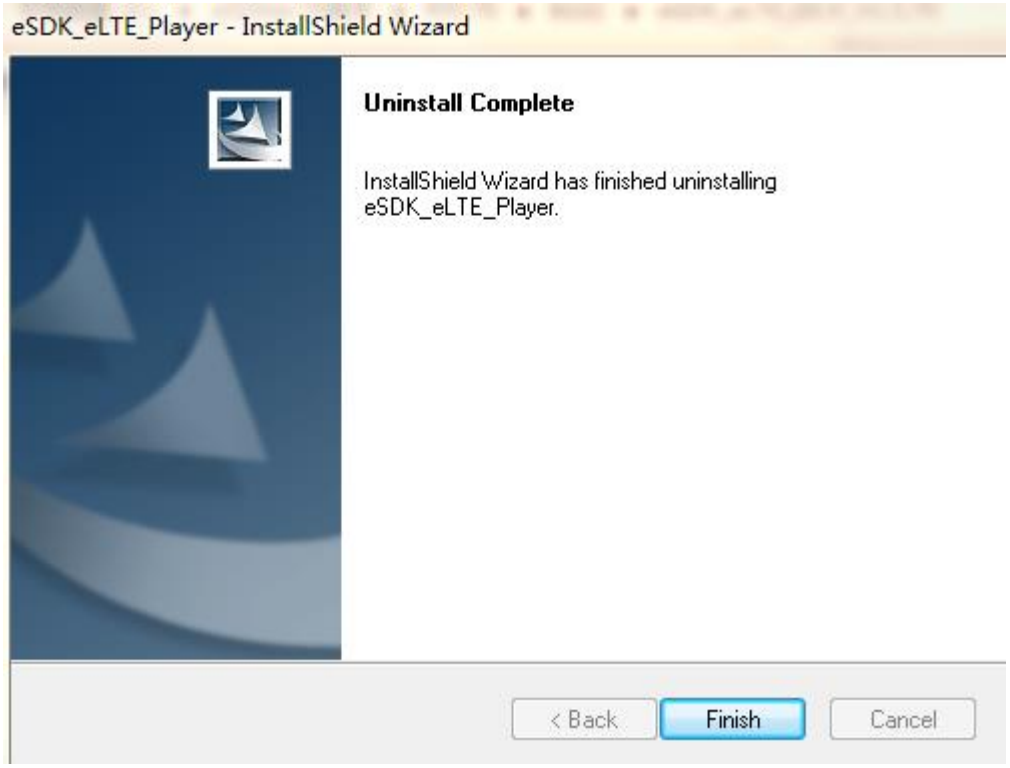
步骤5 提示如下表示安装成功，单击“**Finish**”。



步骤6 如果想卸载服务，则以管理员的身份运行eSDK_eLTE_OCX_V2.1.00.exe，选择“**Remove**”，单击“**Next**”并确认卸载。



步骤7 系统弹出如下图所示提示信息，则表示服务卸载成功。

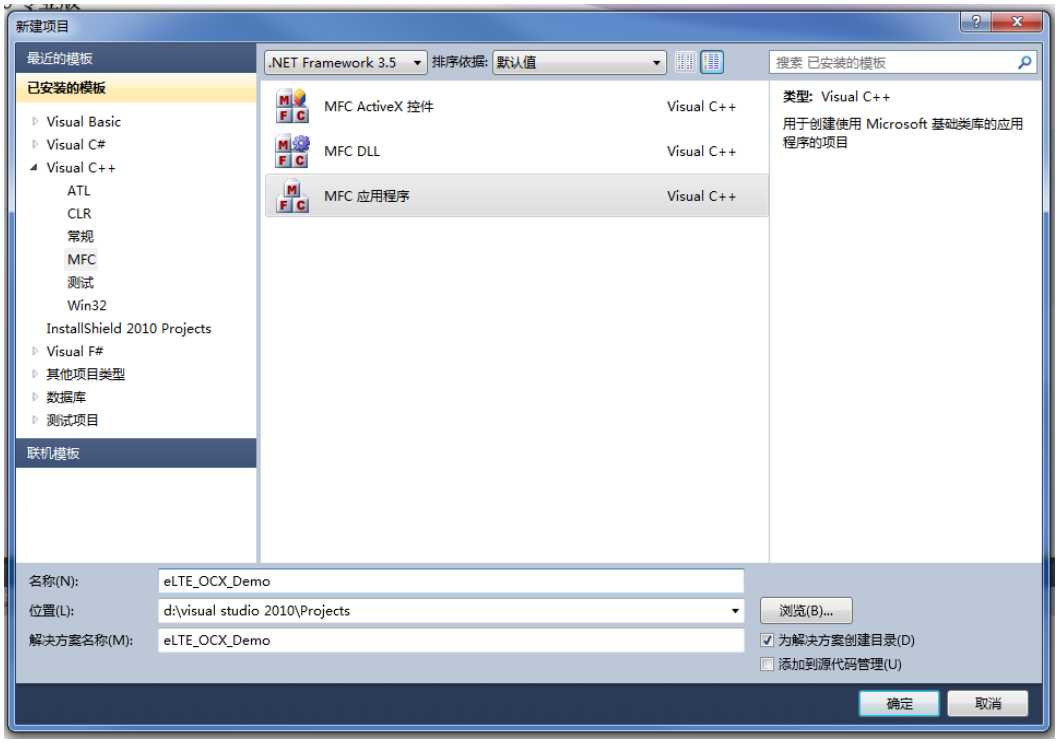


----结束

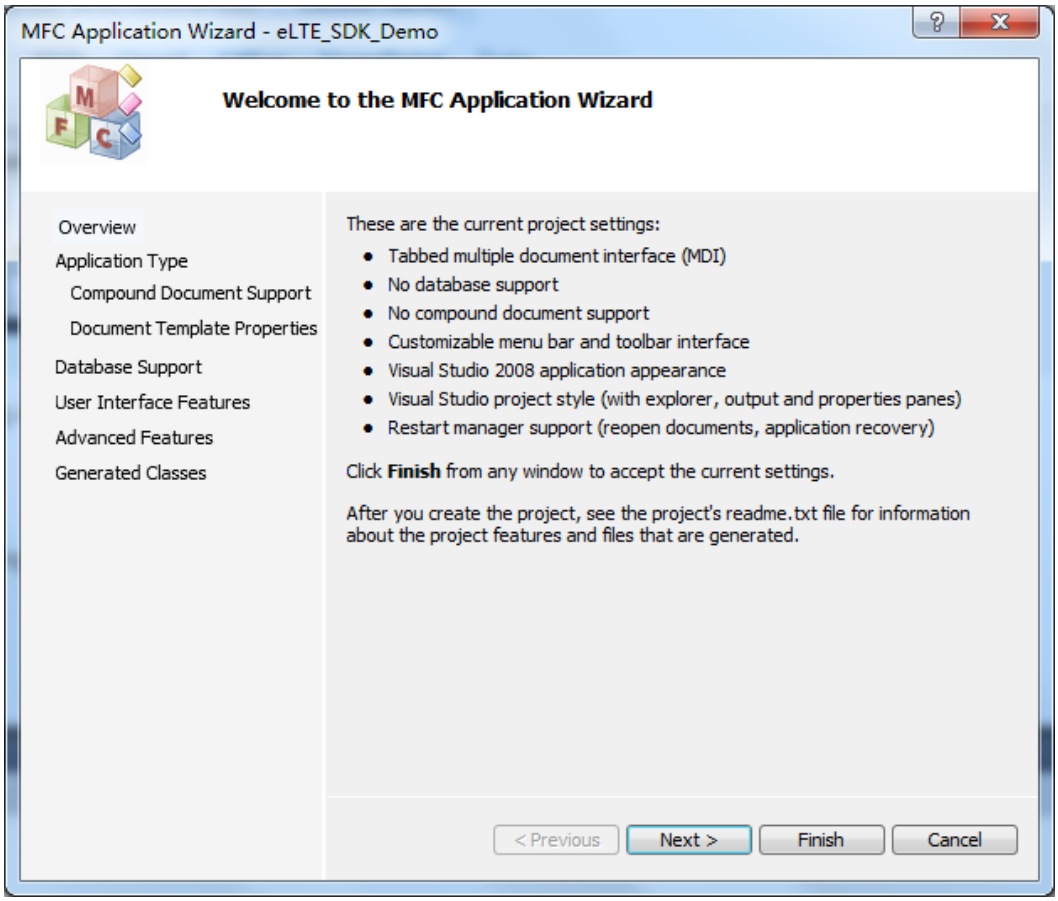
4.3 创建工程

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

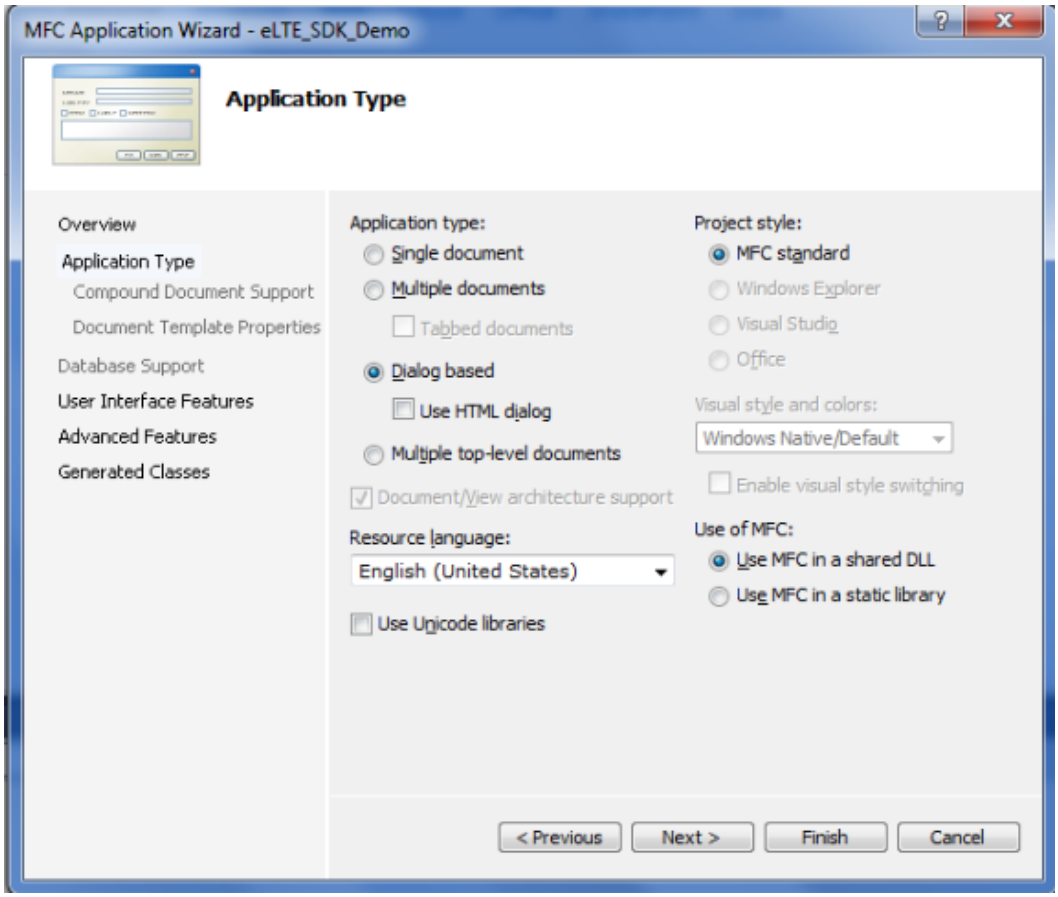
步骤2 选择“**Visual C++ > MFC > MFC Application**”，输入项目名称“eLTE_OCX_Demo”，名称确定后，点击“**确定**”按钮。



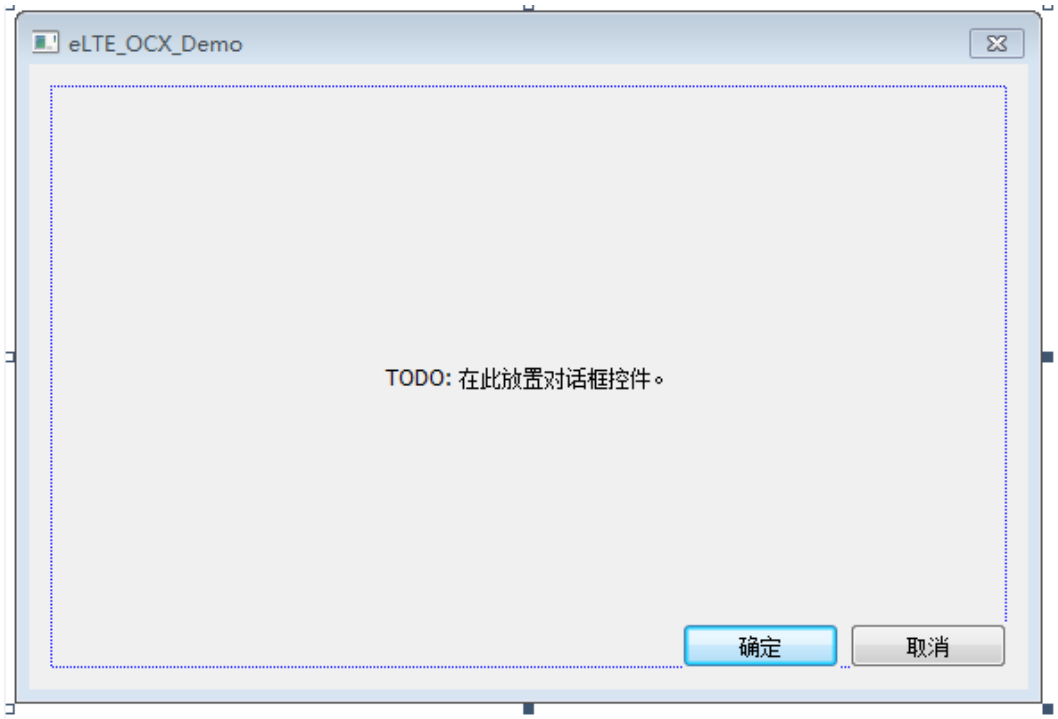
步骤3 单击“**OK**”，进入MFC应用程序向导，系统显示界面如下，点击"**Next >**"。



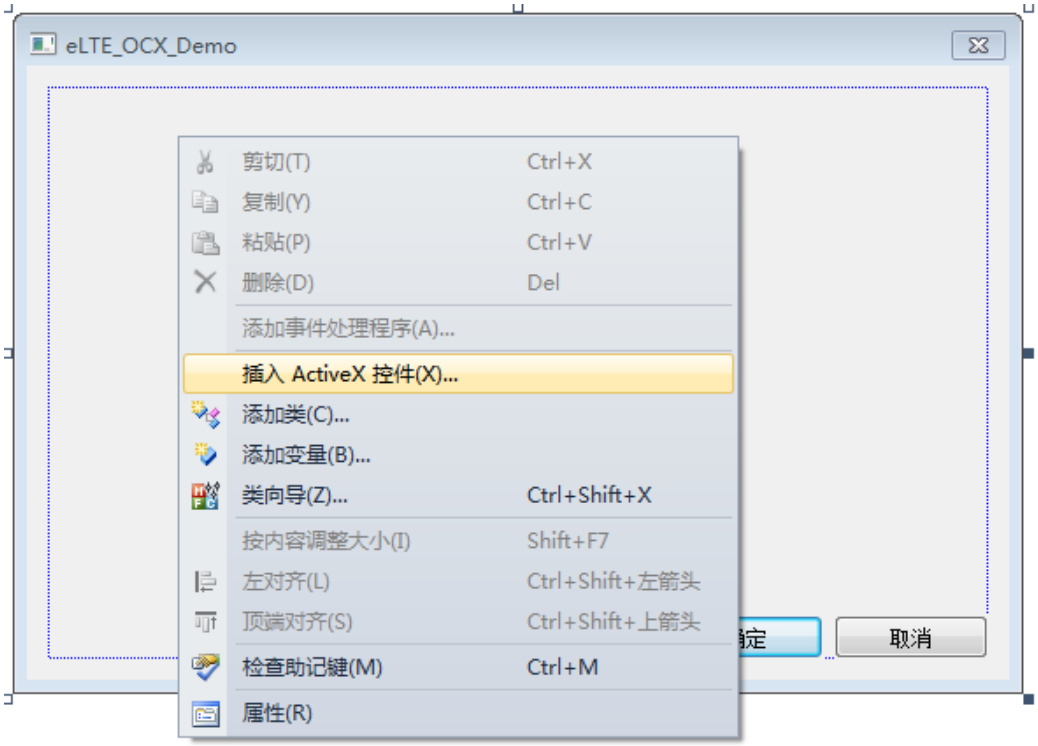
步骤4 单击“**Next**”，进入MFC应用程序类型窗口，选择应用程序类型为“**Dialog based**”，取消勾选“**Use Unicode libraries**”，系统显示界面如下。



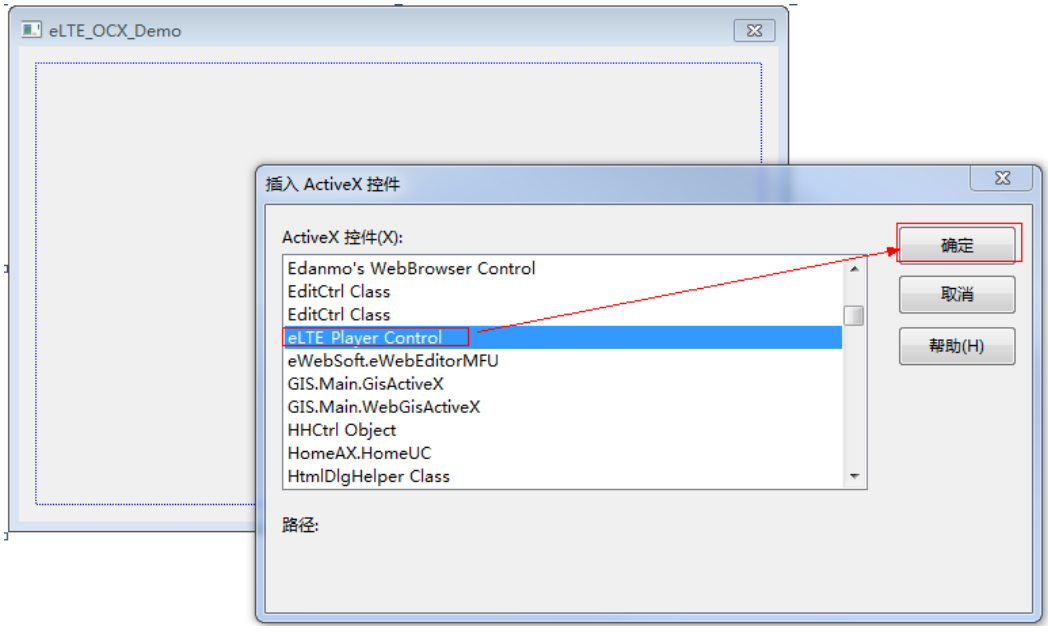
步骤5 单击“**Finish**”，完成新工程的创建。产生工程默认界面如下图所示。



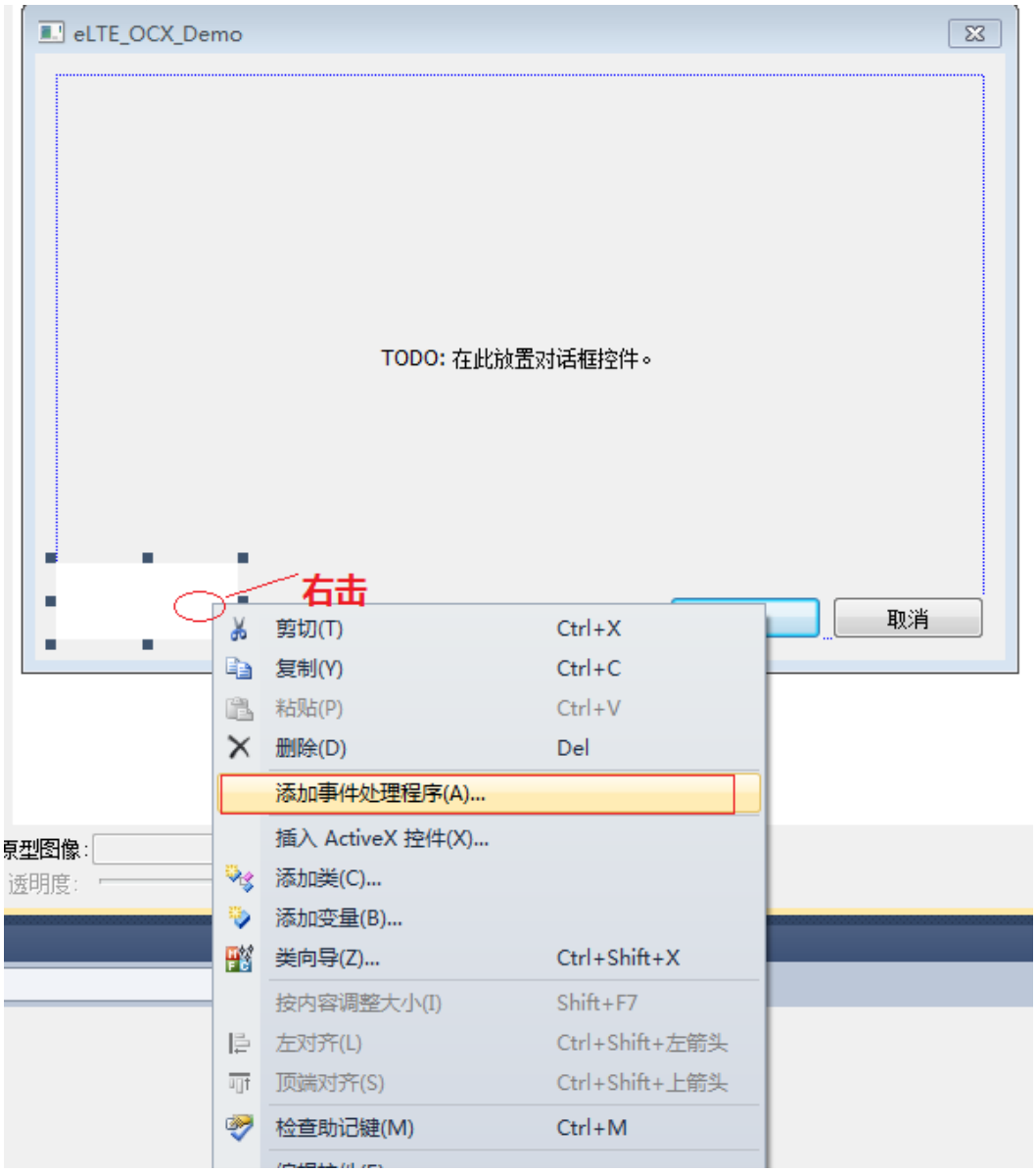
步骤6 在任意空白处单机鼠标右键，插入**ActiveX**控件。



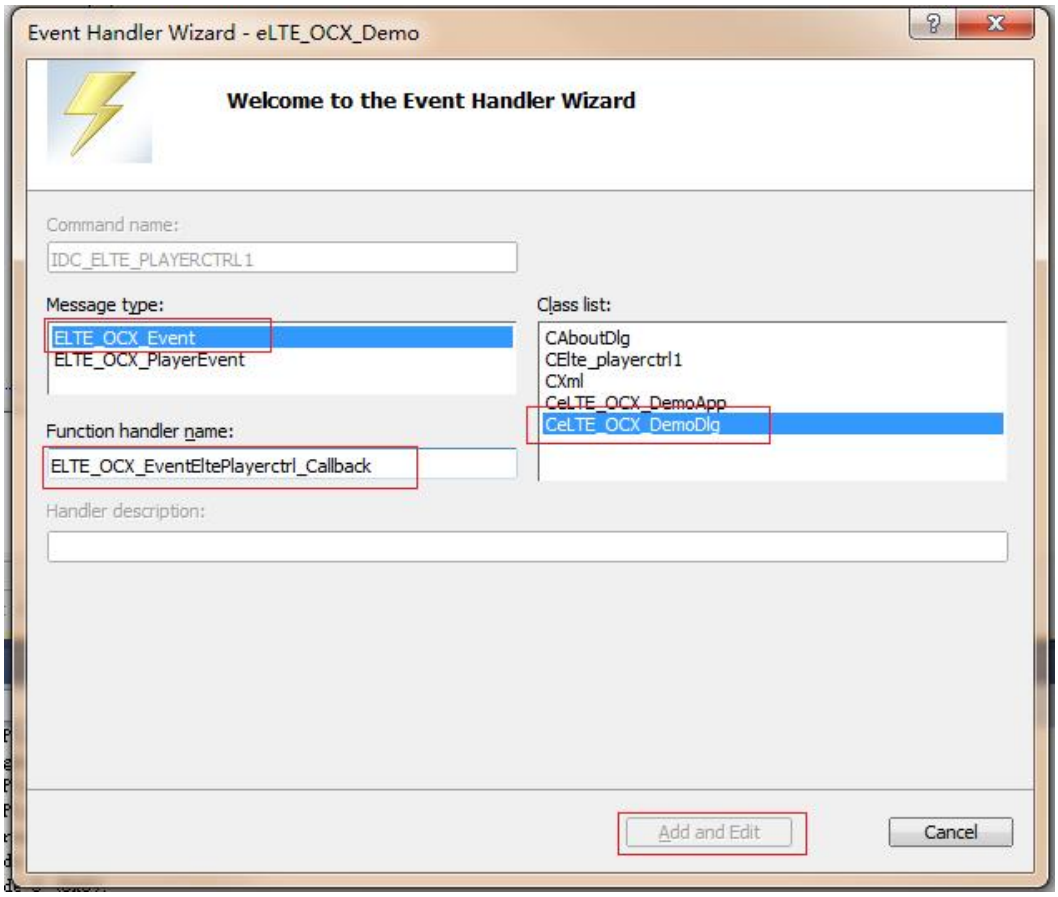
步骤7 选择“**eLTE_Player_Control**”并确认插入，如下图。



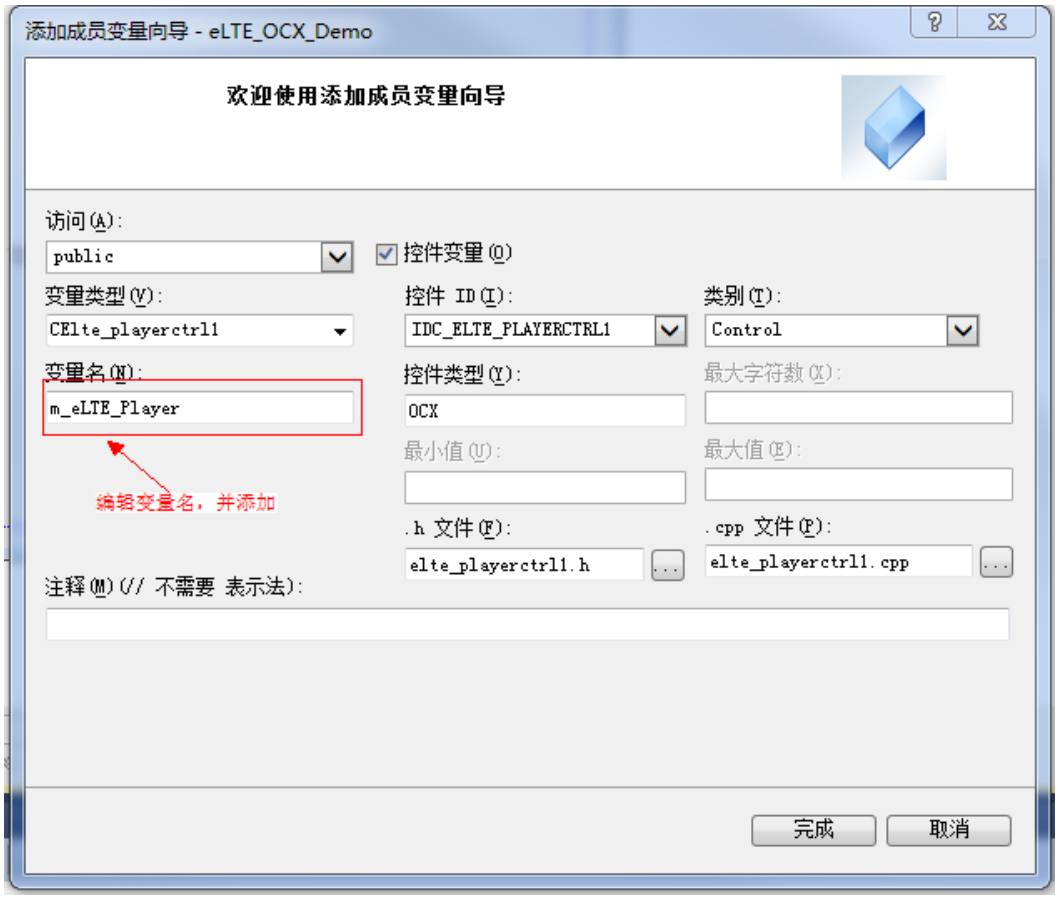
步骤8 插入完成后，如下图，会有白色方框(OCX activeX控件)出现，在其上点击鼠标**右键**，选择**添加事件处理程序**。



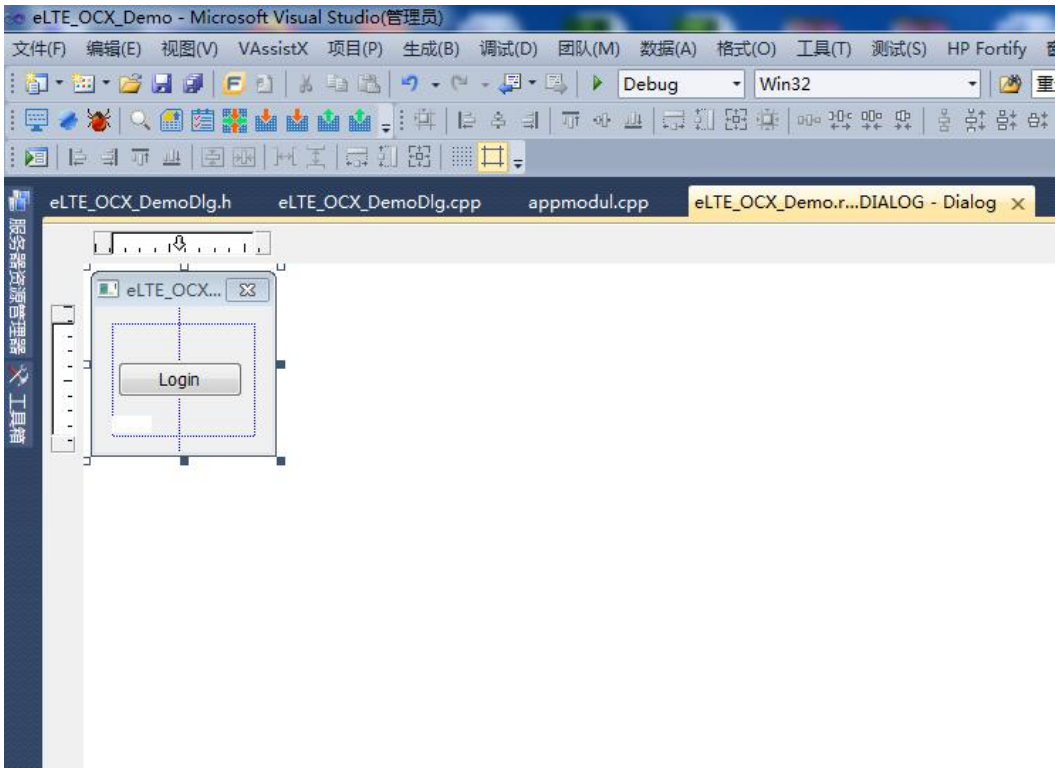
步骤9 确认好后**"添加编辑"**。（增加此函数用以处理回调消息)。



步骤10 在步骤4 白色方框(OCX activeX控件)上单击鼠标**右键**，选择**添加变量**来添加OCX变量。



步骤11 使用**Delete**键删除自动生成的控件(activex控件除外)，添加登录Button控件，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 双击刚才添加的login button控件，在打开的OnBnClickedButtonLogin函数中调用【**ELTE_OCX_Load(加载插件)**】、【**ELTE_OCX_Login(用户登录)**】加载插件和用户登录两个接口，实现登录调度台账号的功能，可以参考接口文档，了解接口调用方法。示例代码如下。

```
//cpp code
//调用加载OCX接口
CString strResult; strResult = m_eLTE_Player.ELTE_OCX_Load(1);
//具体登录信息需要按照实际情况自行配置
//用户名
CString m_strName = "4120";
//登录密码
CString m_strPasswd = "4120";
//server IP
CString m_strServerIP = "172.22.9.120";
//本地IP地址
CString m_strLocalIP = "172.24.4.225";
//端口号
CString m_strSipPort = "5060";
//调用登录接口
strResult = m_eLTE_Player.ELTE_OCX_Login(m_strName, m_strPasswd, m_strServerIP, m_strLocalIP,
m_strSipPort);
```

步骤2 在eLTE_OCX_DemoDlg.cpp源代码中，在void CeLTE_OCX_DemoDlg::ELTE_OCX_EventEltePlayerctrl_CallBack(unsigned long ulEventType, LPCTSTR pEventDataXml)函数中添加对回调事件的处理，只有当收到回调事件EVENT_NOTIFY_RESOURCE _STATUS中，StatusValue值为4011时，才表明登录成功。示例代码如下。

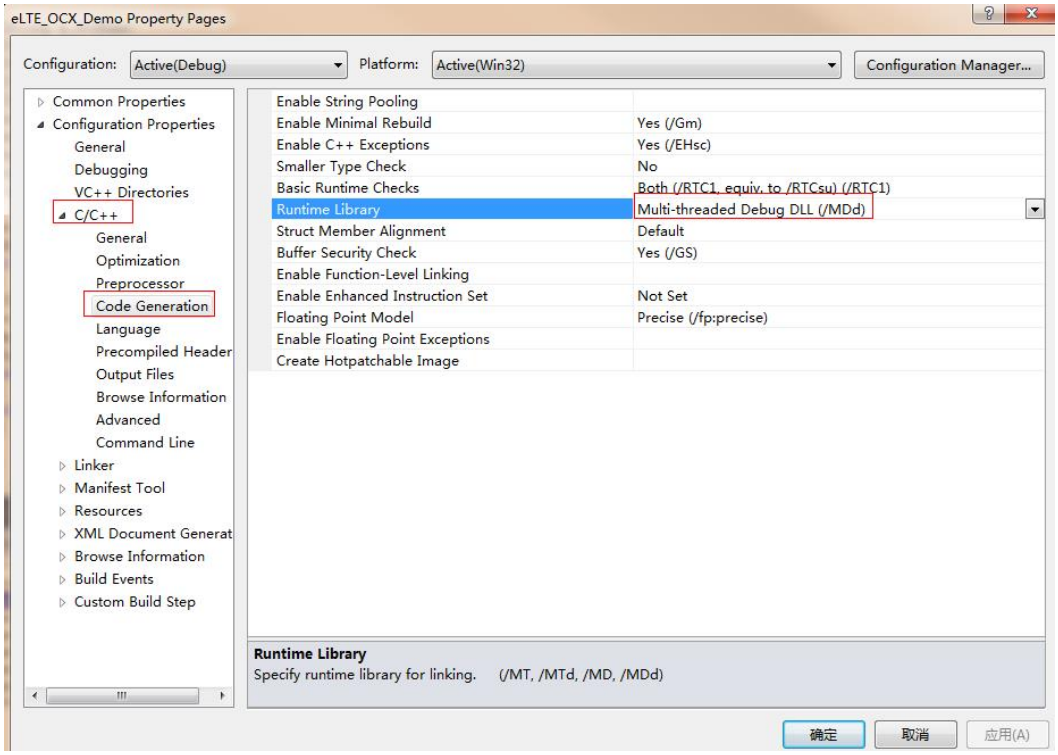
```
//cpp code
//将回调消息赋给str字符串
```

```
CString str=pEventDataXml;
//查找回调消息中是否有注册成功的状态值
int i = str.Find("<StatusValue>4011</StatusValue>");
//判断如果有注册成功的状态值，同时回调事件类型ulEventType = 2 ，则给出登录成功提示。
if(ulEventType==2 && i!=-1)
MessageBox("login success");
```

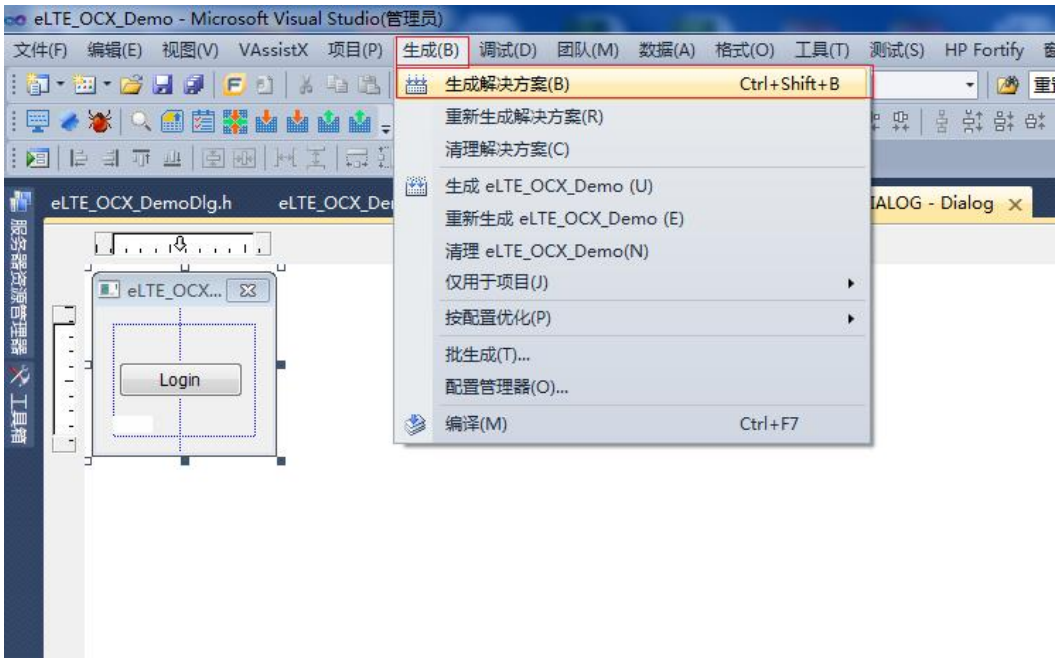
----结束

4.5 编译调试

步骤1 设置工程属性中C/C++中运行库为“**Multi-threaded Debug DLL(/MDd)**”。



步骤2 点击**Build** > **Rebuild Solution** 生成.exe文件，系统显示界面如下。



步骤3 若您已购买华为eLTE产品，请直接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。

在输入框内输入正确的用户名、密码、服务器IP、本机IP以及服务器端口。

点击“**Login**”，提示框显示“login sucess”，表示登录成功。如下图所示。

----结束

5 初始化配置

您需要完成以下初始化配置后再进行视频监控业务二次集成开发。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

OCX会默认在运行程序下面生成相应的日志文件，eSDK日志文件用于查看OCX运行信息，协助问题定位。

如果您需要指定日志路径可以调用设置日志路径【**ELTE_OCX_SetLogPath(设置日志路径)**】接口来实现。

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OCX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_OCX_Load(加载插件)**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
//调用设置日志路径接口，日志路径中“\”必须进行转义
CStringstrRet = m_eLTE_Player.ELTE_OCX_SetLogPath("D:\\log\\");
CString strBegin("<ResultCode>");
CString strEnd("</ResultCode>");
strRet = strRet.Left(strRet.Find(strEnd));
strRet = strRet.Right(strRet.GetLength() -strBegin.GetLength() - strRet.Find(strBegin));
if("0" == strRet)
{
    //接口调用成功
}
```

此外，您也可以通过修改配置文件来设置日志路径和级别，修改C:\eSDK_eLTE\eLTE_SDK_Player路径下eSDKClientLogCfgOcx.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 设置媒体面【可选】

调用【**ELTE_OCX_SetBypassBuildMedia(设置媒体面)**】接口设置媒体面，若不设置则默认不使用eLTE SDK媒体面。

注意

1.若使用本地音频如语音呼叫或视频伴音，必须设置使用媒体面。

2.入参0表示使用媒体面，入参1表示不使用媒体面。

代码示例

```
//cpp code
//设置使用媒体面
CString strRet = m_eLTE_Player.ELTE_OCX_SetBypassBuildMedia (0);
CString strBegin("<ResultCode>");
CString strEnd("</ResultCode>");
strRet = strRet.Left(strRet.Find(strEnd));
strRet = strRet.Right(strRet.GetLength() -strBegin.GetLength() - strRet.Find(strBegin));
if("0" == strRet)
{
```

```
    //接口调用成功
}
```

5.3 加载 OCX 控件

调用【**ELTE_OCX_Load(加载插件)**】接口，初始化OCX控件资源、预分配内存等操作。

说明

在使用OCX插件提供的相关功能之前，需要调用此接口加载OCX插件。

1. OCX初始化的时候，插件加载类型必须为ulType=1，这样业务层才可正常工作。当需要开启多个媒体窗口的时候，需要重新滴啊用load函数，此时参数ulType=2。
2. 插件加载类型**ulType**在一个程序中只能调用一次，即使在**ELTE_OCX_UnLoad(卸载插件)**之后再调用也无效，只能通过重新启动程序。

代码示例如下。

```
//cpp code
//调用加载控件接口
CStringstrRet = m_eLTE_Player.ELTE_OCX_Load (1);
CString strBegin("<ResultCode>");
CString strEnd("</ResultCode>");
strRet = strRet.Left(strRet.Find(strEnd));
strRet = strRet.Right(strRet.GetLength() -strBegin.GetLength() - strRet.Find(strBegin));
if("0" == strRet)
{
    //接口调用成功
}
```

5.4 获取 OCX 版本号【可选】

调用【**ELTE_OCX_GetVersion(获取插件版本号)**】接口，获取OCX版本号。代码示例如下。

```
//cpp code
CStringstrRet = m_eLTE_Player.ELTE_OCX_GetVersion(1);
CString strBegin("<ResultCode>");
CString strEnd("</ResultCode>");
strRet = strRet.Left(strRet.Find(strEnd));
strRet = strRet.Right(strRet.GetLength() -strBegin.GetLength() - strRet.Find(strBegin));
if("0" == strRet)
{
    //接口调用成功
}
```

5.5 设置消息回调事件函数

OCX控件的消息回调事件函数可通过**加载OCX控件的第8步**，添加事件处理程序来完成自动添加，消息处理格式如下：

例：用户状态消息回调

```
//cpp code
#define GET_XML_ELEM_VALUE_STR(xml, elem) \
  CXml::Instance().XmlParseElemValue(xml, elem)
//XML字串解析函数
CString CXml::XmlParseElemValue(const CString& xmlStr, const CString& elem)
{
  CString value(xmlStr);
  CString elemBegin(elem);
```

```
  CString elemEnd(elem);
  elemBegin.Insert(0, _T("<"));
  elemBegin.Append(_T(">"));
  elemEnd.Insert(0, _T("</"));
  elemEnd.Append(_T(">"));
  value = value.Left(value.Find(elemEnd));
  value = value.Right(value.GetLength() - elemBegin.GetLength() - value.Find(elemBegin));
  return value;
}
//事件回调处理函数
void CeLTE_PlayerDemoDlg::ELTE_OCX_EventEltePlayerctrl1(unsigned long ulEventType, LPCTSTR
pEventDataXml)
{
  //判断消息回调类型
  switch (ulEventType)
  {
  case EVENT_NOTIFY_USER_STATUS:  // notifyUserStatus
    {

      CString strUserID = GET_XML_ELEM_VALUE_STR(pEventDataXml, _T("UserID"));
      int iStatusType = GET_XML_ELEM_VALUE_INT(pEventDataXml, _T("StatusType"));
      int iStatusValue = GET_XML_ELEM_VALUE_INT(pEventDataXml, _T("StatusValue"));
      m_DcDlg.UpdateUserStatus(strUserID, iStatusValue);
      // 显示视频回传当前状态
      CString strEventMsg;
      strEventMsg.Format(_T("UserID:%s Type:%d Value:%d 【"), strUserID, iStatusType,
iStatusValue);
      //当前状态判断
      if (iStatusValue == 4011)
      {
        //在线状态
        strEventMsg.Append(_T("online"));
      }
      else if (iStatusValue == 4012)
      {
        //离线状态
        strEventMsg.Append(_T("offline"));
      }
      else if (iStatusValue == 4020)
      {
        //开启点呼
        strEventMsg.Append(_T("start calling"));
      }
      else if (iStatusValue == 4021)
      {
        //响铃
        strEventMsg.Append(_T("Ringing"));
      }
      else if (iStatusValue == 4022)
      {
        //使用中
        strEventMsg.Append(_T("under use"));
      }
      else if (iStatusValue == 4023)
      {
        //空闲
        strEventMsg.Append(_T("free"));
      }
      strEventMsg.Append(_T("】\r\n"));
      strEventMsg.Insert(0,GetTimeString());
      m_DcDlg.m_strEvent0.Append(strEventMsg);
    }
    break;
    case EVENT_NOTIFY_P2P_VIDEO_CALL_STATUS:  // notifyP2PCallStatus
    ......
  }
}
```

说明

事件类型参考:

EVENT_NOTIFY_USER_STATUS // 设备状态变化事件通知
EVENT_NOTIFY_P2P_VIDEO_CALL_STATUS // 开始实况事件通知
EVENT_NOTIFY_RESOURCE_STATUS // 群组关系状态变化事件通知
EVENT_NOTIFY_PROVISION_CHANGE // 设备属性配置变更通知事件
EVENT_NOTIFY_PROVISION_ALLRESYNC // 自动下载配置数据通知事件
EVENT_NOTIFY_P2P_CALL_STATUS // 点呼状态变化事件
EVENT_NOTIFY_GROUP_STATUS // 组呼状态变化事件
EVENT_NOTIFY_MODULE_STATUS // 模块组件状态事件
EVENT_NOTIFY_GIS_REPORT // 终端GIS信息上报事件
EVENT_NOTIFY_GIS_STATUS // 终端订阅状态上报事件
EVENT_NOTIFY_SDS_REPORT // 短信上报事件
EVENT_NOTIFY_SDS_RETCODE // 短信发送是否成功事件

5.6 用户登录 eAPP 服务器

调用用户登录【**ELTE_OCX_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。

```
//cpp code
//调用登录接口，具体的登录信息需要根据实际环境进行修改
CStringstrRet = m_eLTE_Player.ELTE_OCX_Login ("4120","4120", "172.22.9.105","172.24.4.253",
"5060");
CString strBegin("<ResultCode>");
CString strEnd("</ResultCode>");
strRet = strRet.Left(strRet.Find(strEnd));
strRet = strRet.Right(strRet.GetLength() -strBegin.GetLength() - strRet.Find(strBegin));
if("0" == strRet)
{
    //接口调用成功
}
```

注意

调度台密码属于敏感信息，发布给客户使用的版本代码和日志中不要明文保存用户的密码等安全信息。

如需使用获取录像文件列表功能，必须在登录成功后调用初始化媒体服务器【**ELTE_OCX_ProvisionManagerInitMRS(初始化媒体服务器)**】接口。当有多个多媒体录制播放服务器（MRS）时，可以多次调用该接口逐个初始化。

```
//cpp code
//调用初始化媒体服务器接口，对于一体化调度机，一般eMDC的IP与MRSIP相同，对于分布式组网方式，一般
MRSIP与eMDCIP不通，也可能存在多个MRSIP
CStringstrRet = m_eLTE_Player.ELTE_OCX_ProvisionManagerInitMRS ("172.22.9.105");
CString strBegin(<ResultCode>");
CString strEnd("</ResultCode>");
strRet = strRet.Left(strRet.Find(strEnd));
strRet = strRet.Right(strRet.GetLength() -strBegin.GetLength() - strRet.Find(strBegin));
if("0" == strRet)
{
    //接口调用成功
}
```

5.7 触发状态上报

用户应用程序中怎么检测资源处于什么状态呢？那么就需要我们在应用程序启动后，调用触发状态上报接口，业务事件上报或者变更上报等信息会在业务进行时通过消息回调事件函数上报。

在调用触发状态上报接口之前，必须收到【**EVENT_NOTIFY_PROVISION_ALLRESYNC(自动下载配置数据通知事件通知)**】的回调消息，否则调用各种类型的查询接口返回值为空。

调用触发状态上报【**ELTE_OCX_TriggerStatusReport(触发状态上报)**】接口，开启状态上报。当应用程序启动后，事件上报机制会被启动。业务事件上报或者变更上报等信息会在对应业务进行时通过各自的事件上报通知函数上报。

代码样例如下：

```
//cpp code
CStringstrRet = m_eLTE_Player.ELTE_OCX_TriggerStatusReport (1);
CString strBegin("<ResultCode>");
CString strEnd("</ResultCode>");
strRet = strRet.Left(strRet.Find(strEnd));
strRet = strRet.Right(strRet.GetLength() -strBegin.GetLength() - strRet.Find(strBegin));
if("0" == strRet)
{
    //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

6.1.1 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。

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。

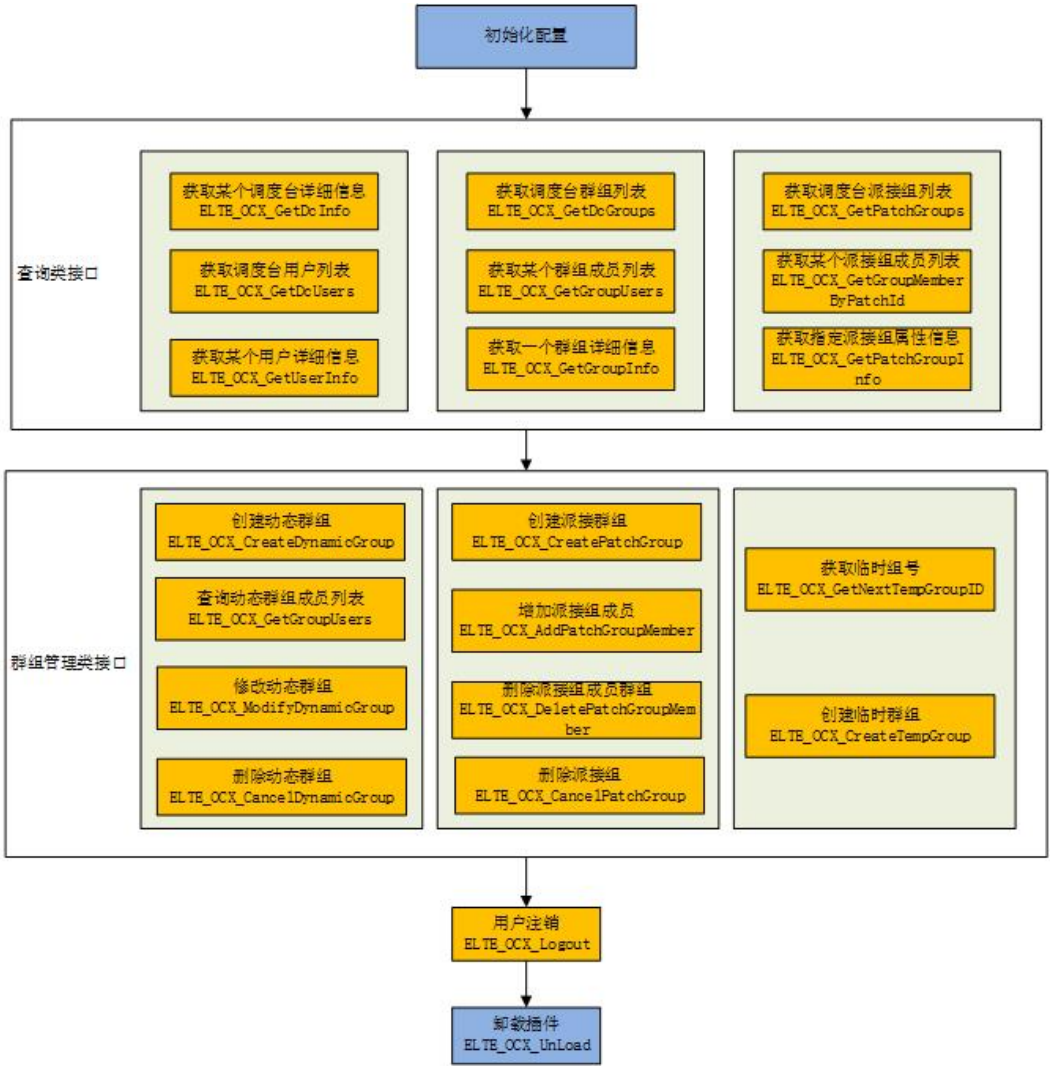
我们将按照如下场景进行详细的分析和介绍：

- **用户和群组管理**
- **获取录音录像文件列表**

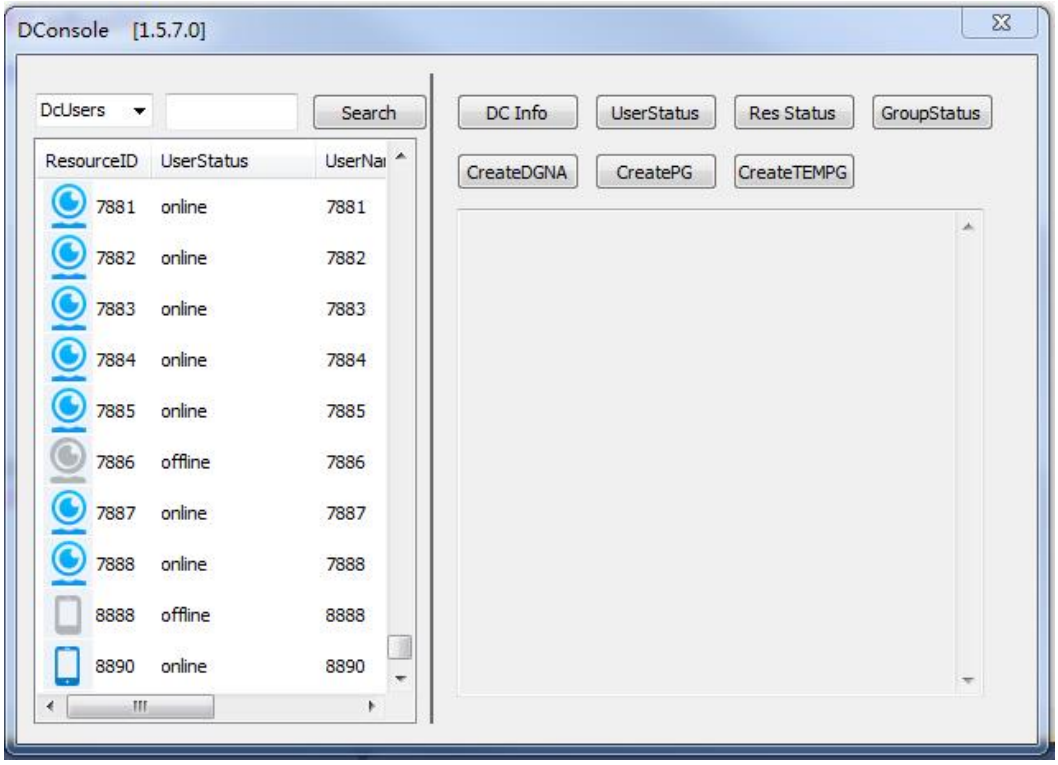
6.1.2 用户和群组管理

本章节的开发任务：获取调度台信息，获取调度台用户列表，获取某个用户的详细信息，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

已完成**初始化配置**。

获取某个调度台的详细信息

调用【**ELTE_OCX_GetDcInfo(获取某个调度台的详细信息)**】接口，获取当前登录的调度台详细信息，接口入参是当前调度台ID。

接口调用代码示例如下：

```
//cpp code
//获取调度台4116信息
CString strResult = m_eLTE_Player.ELTE_OCX_GetDcInfo("4116");
```

strResult返回信息如下：

```
//xml code
//获取调度台返回信息
<Content>
    <ResultCode>0</ResultCode>
    <DcInfo>
        <DcID>4116</DcID>
        <Priviledge>32767</Priviledge>
        <Role>1</Role>
        <Alias/>
    </DcInfo>
</Content>
```

ResultCode=0,表示接口调用成功，DcInfo节点包含该调度台的DcID、Priviledge、Role、Alias信息，您可根据开发需求对这些信息进行处理和显示。

获取调度台成员列表

调用【**ELTE_OCX_GetDcUsers(获取调度台成员列表)**】接口，获取调度台可以管理的所有用户列表，接口入参只能是当前调度台ID。

接口调用代码示例如下：

```
//cpp code
//获取当前调度台4116的用户列表
CString strResult = m_eLTE_Player.ELTE_OCX_GetDcUsers("4116");
```

strResult返回信息如下。

```
//xml code
//获取调度台用户列表返回信息
<Content>
    <ResultCode>0</ResultCode>
    <UserInfoList>
        <UserInfo>
            <UserID>4001</UserID>
            <UserCategory>9</UserCategory>
            <UserPriority>15</UserPriority>
            <UserName>4001</UserName>
        </UserInfo>
        ......
    </UserInfoList>
</Content>
```

ResultCode=0,表示接口调用成功，UserInfoList节点包含UserID、UserCategory、UserPriority、UserName信息，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。

获取某个用户的详细信息

调用【**ELTE_OCX_GetUserInfo(获取某个用户的详细信息)**】接口，获取调度员所管辖的某个指定成员的详细信息，包括用户ID、用户类别、用户优先级、用户名称等。

接口调用代码示例如下：

```
//cpp code
//获取终端用户8890的详细信息
CString strResult = m_eLTE_Player.ELTE_OCX_GetUserInfo("8890");
```

strResult返回信息如下：

```
//xml code
<Content>
    <ResultCode>0</ResultCode>
    <UserInfo>
        <UserID>8890</UserID>
        <UserCategory>9</UserCategory>
        <UserPriority>15</UserPriority>
        <UserName>8890</UserName>
    </UserInfo>
</Content>
```

ResultCode=0,表示接口调用成功，UserInfo节点包含UserID、UserCategory、UserPriority、UserName信息。

获取调度台群组列表

调用【**ELTE_OCX_GetDcGroups(获取调度台群组列表)**】接口，获取调度台可以管理的所有群组列表，接口入参只能是当前调度台ID。

接口调用代码示例如下：

```
//cpp code
//获取当前调度台4116的群组列表
CString strResult = m_eLTE_Player.ELTE_OCX_GetDcGroups("4116");
```

strResult返回信息如下：

```
//xml code
<Content>
    <ResultCode>0</ResultCode>
    <GroupInfoList>
        <GroupInfo>
            <GroupID>1001</GroupID>
            <GroupCategory>1</GroupCategory>
            <GroupPriority>15</GroupPriority>
            <GroupName>测试群组</GroupName>
        </GroupInfo>
        ......
    </GroupInfoList>
</Content>
```

ResultCode=0表示接口调用成功，GroupInfoList节点包含GroupID、GroupCategory、GroupPriority、GroupName信息，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_OCX_GetGroupUsers(获取某个群组的成员列表)**】接口，获取某个群组的成员列表。

接口调用代码示例如下：

```
//cpp code
//获取群组1001的成员列表
strResult = m_eLTE_Player.ELTE_OCX_GetGroupUsers("1001");
```

strResult返回信息如下：

```
//xml code
<Content>
    <ResultCode>0</ResultCode>
    <GroupUserInfoList>
        <GroupUserInfo>
            <UserID>8888</UserID>
            <GroupID>1001</GroupID>
            <UserPriorityInGroup>15</UserPriorityInGroup>
            <MemberType>1</MemberType>
        </GroupUserInfo>
        ......
    </GroupUserInfoList>
</Content>
```

GroupUserInfo节点包含了该调度台的UserID、GroupID、UserPriorityInGroup、MemberType信息，其中成员类别0表示动态重组成员，1表示普通用户成员。

获取某个群组详细信息

调用【**ELTE_OCX_GetGroupInfo(获取某个群组的详细信息)**】接口，获取某个群组的详细信息。

接口调用代码示例如下：

```
//cpp code
//获取群组1001的详细信息
CString strResult = m_eLTE_Player.ELTE_OCX_GetGroupInfo("1001");
```

strResult返回信息如下。

```
//xml code
<Content>
    <ResultCode>0</ResultCode>
    <GroupInfo>
        <GroupID>1001</GroupID>
        <GroupCategory>1</GroupCategory>
        <GroupPriority>15</GroupPriority>
        <GroupName>TEST1001</GroupName>
        <GroupCreator></GroupCreator>
    </GroupInfo>
</Content>
```

GroupInfo节点包含了该调度台的GroupID、GroupCategory、GroupPriority、GroupName、GroupCreator信息。

获取调度台派接组列表

调用【**ELTE_OCX_GetPatchGroups(获取某个调度台的所有派接组列表)**】接口，获取DC调度台可以管理的所有派接组列表。

接口调用代码示例如下：

```
//cpp code
//获取当前调度台4116的用户列表
CString strResult = m_eLTE_Player.ELTE_OCX_GetPatchGroups("4116");
```

strResult返回信息如下：

```
//xml code
<Content>
    <ResultCode>0</ResultCode>
    <PatchGroupInfoList>
        <PatchGroupInfo>
            <GroupNumber>99899999</GroupNumber>
            <SetupDcId>4130</SetupDcId>
            <PGPriority>3</PGPriority>
            <DcPatchIndex>1</DcPatchIndex>
            <PGName>a1</PGName>
        </PatchGroupInfo>
        ......
    </PatchGroupInfoList>
</Content>
```

PatchGroupInfo节点包含了该调度台的派接组ID、创建者ID、优先级、派接组索引号等信息。

获取某个派接组的成员列表

调用【**ELTE_OCX_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。

```
//cpp code
//调用获取派接组99899999的成员列表接口
CString strResult = m_eLTE_Player.ELTE_OCX_GetGroupMemberByPatchId ("99899999");
```

strResult返回信息如下：

```
//xml code
<Content>
    <ResultCode>0</ResultCode>
    <PatchGroupMemberList>
        <PatchGroupMember>
            <GroupNumber>99899999</GroupNumber>
            <MemberGroupId>1001</MemberGroupId>
        </PatchGroupMember>
        ......
    </PatchGroupMemberList>
</Content>
```

获取指定派接组属性信息

调用【**ELTE_OCX_GetPatchGroupInfo(获取指定派接组属性信息)**】接口，获取指定派接组属性信息。

接口调用代码示例如下：

```
//cpp code
//调用获取派接组99899999的属性信息
CString strResult = m_eLTE_Player.ELTE_OCX_GetPatchGroupInfo ("99899999");
```

strResult返回信息如下：

```
//xml code
<Content>
    <ResultCode>0</ResultCode>
    <PatchGroupInfo>
        GroupNumber>99899999</GroupNumber>
        <SetupDcId>4130</SetupDcId>
```

```
        <PGPriority>3</PGPriority>
        <DcPatchIndex>1</DcPatchIndex>
        <PGName>a1</PGName>
        <VPNID>0</VPNID>
    </PatchGroupInfo>
</Content>
```

创建动态群组

调用【**ELTE_OCX_CreateDynamicGroup(创建动态组)**】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。

接口调用代码示例如下。

```
//cpp code
//定义和构造strDGNAParam字符串，作为创建动态组接口入参
CString strDGNAParam;
strDGNAParam.Append("<Content>");
//动态群组ID，不填写自动生成的动态组组号
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>");
//该组别名，字符串长度小于等于32
strDGNAParam.Append("<Alias>");
strDGNAParam.Append("test");
strDGNAParam.Append("</Alias>");
//该动态组优先级，取值范围为1～15
strDGNAParam.Append("<Priority>");
strDGNAParam.Append("15");
strDGNAParam.Append("</Priority>");
//该组的最大通话时长，取值范围为1～66535
strDGNAParam.Append("<MaxPeriod>");
strDGNAParam.Append("60");
strDGNAParam.Append("</MaxPeriod>");
//群组成员列表，最大值8个
strDGNAParam.Append("<GroupList>");
//群组成员ID
strDGNAParam.Append("<GroupID>");
strDGNAParam.Append("1001");
strDGNAParam.Append("</GroupID>");
strDGNAParam.Append("</GroupList>");
//用户成员列表，最大值200个
strDGNAParam.Append("<UserList>");
//用户成员ID
strDGNAParam.Append("<UserID>");
strDGNAParam.Append("4114");
strDGNAParam.Append("</UserID>");
strDGNAParam.Append("</UserList>");
strDGNAParam.Append("</Content>");
//调用创建派接组接口
CString strResult = m_eLTE_Player.ELTE_OCX_CreateDynamicGroup(strDGNAParam);
//判断派接组调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

插件等待接收调度机返回结果，如果失败直接返回失败原因；成功返回0，调度开始处理，插件通过消息回调事件函数【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】获取调度机返回的执行结果，如下：

```
//xml code
<Content>
    <ResourceID>99899986</ResourceID>
    <ResourceName></ResourceName>
    <StatusType>11</StatusType>
```

```
    <StatusValue>4003</StatusValue>
    <AttachingGroup>0</AttachingGroup>
</Content>
```

ResourceID=动态群组号，StatusValue=4003资源指派状态，说明动态群组创建成功,动态群组创建成功后会自动订阅并加入群组。

获取动态群组成员列表

调用【**ELTE_OCX_GetGroupUsers(获取某个群组的成员列表)**】接口，可参考获取一个群组成员列表。

修改动态群组

调用【**ELTE_OCX_ModifyDynamicGroup(修改动态群组)**】接口，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。
2. 调用该接口前，可通过调用【**ELTE_OCX_GetGroupUsers(获取某个群组的成员列表)**】接口，从该列表中获取待修改的动态群组成员信息。

调用修改动态群组接口代码示例如下：

```
//cpp code
//定义修改动态群组入参字符串
CString strParam;
//构造入参字符创各节点
strParam.Append("<Content>");
strParam.Append("<DynamicGroupInfo>");
//动态群组号码
strParam.Append("<GroupID>");
strParam.Append("99899999");
strParam.Append("</GroupID>");
strParam.Append("<AddUserList>");
//添加的用户ID，可添加多个，不要与原来的重复
strParam.Append("<AddUserID>");
strParam.Append("8890");
strParam.Append("</AddUserID>");
strParam.Append("</AddUserList>");
strParam.Append("<DeleteUserList>");
strParam.Append("<DeleteUserID>");
strParam.Append("4115");
strParam.Append("</DeleteUserID>");
strParam.Append("</DeleteUserList>");
strParam.Append("</DynamicGroupInfo>");
strParam.Append("</Content>");
//调用修改动态群组接口
CString strResult =m_eLTE_Player.ELTE_OCX_ModifyDynamicGroup("4121", strParam);
//判断接口是否调用成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

通过解析消息回调事件函数【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】，得到xml消息格式如下：

```
//xml code
<Content>
    <ResourceID>0</ResourceID>
    <ResourceName></ResourceName>
    <StatusType>21</StatusType>
```

```
    <StatusValue>4024</StatusValue>
    <AttachingGroup>0</AttachingGroup>
    <Cause>0</Cause>
</Content>
```

如果StatusType=21，StatusValue=4024，说明修改动态群组成功。

删除动态群组

调用【**ELTE_OCX_CancelDynamicGroup(删除动态组)**】接口，可根据资源ID（动态组号ID）删除动态组。

说明

1. 只有动态群组的创建者才能删除该动态群组。
2. 删除动态群组时，该动态群组必须处于订阅状态，可调用【**ELTE_OCX_SubJoinGroup(订阅并自动加入群组)**】订阅该群组。

删除动态群组代码示例如下。

```
//cpp code
//调用删除动态群组接口，删除60001动态群组
CString strRet =m_eLTE_Player.ELTE_OCX_CancelDynamicGroup ("60001");
//判断返回结果码是否成功，ResultCode=0即成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

若收到消息回调事件函数【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】中StatusValue=4004，说明动态群组删除成功。

创建临时群组

调用【**ELTE_OCX_CreateTempGroup(创建临时组)**】接口创建动态群组之前必须先调用【**ELTE_OCX_GetNextTempGroupID(获取临时组号)**】获取临时群组号。

获取临时组号接口调用代码示例如下。

```
//cpp code
//获取临时组号入参空
CString strResult = m_eLTE_Player.ELTE_OCX_GetNextTempGroupID();
```

获取临时组号成功strResult返回信息如下。

```
//xml code
<Content>
    <NextTempGroupID>60009</NextTempGroupID>
</Content>
```

您需要先提取xml字符串中的NextTempGroupID作为临时组号，并赋值给strTempGroupID，首先定义一个函数解析xml中对应节点的值，如下。

```
//cpp code
CString XmlParseElemValue(const CString& xmlStr, const CString& elem)
{
    CString value(xmlStr);
    CString elemBegin(elem);
    CString elemEnd(elem);
    elemBegin.Insert(0, "<");
    elemBegin.Append(">");
    elemEnd.Insert(0, "</");
    elemEnd.Append(">");

    value = value.Left(value.Find(elemEnd));
```

```
    value = value.Right(value.GetLength() - elemBegin.GetLength() - value.Find(elemBegin));

    return value;
}
```

然后调用这个函数提取获取临时群组号strResult返回信息中的GroupID。

```
//cpp code
//解析strResult中NextTempGroupID字段值，并将其赋值给strTempGroupID变量。
CString strTempGroupID = XmlParseElemValue(strResult,NextTempGroupID);
```

临时群组号已经获得，下面我们就可以调用创建动态群组接口了。

调用创建临时组接口代码示例如下。

```
//cpp code
//创建一个临时群组
//定义并构造strDGNAParam字符串，作为创建临时群组入参
CString strDGNAParam;
strDGNAParam.Append("<Content>");
//临时群组号，通过ELTE_OCX_GetNextTempGroupID(获取临时组号)获取
strDGNAParam.Append("<GroupID>");
strDGNAParam.Append(strTempGroupID); //解析xml字串strResult所获取的值
strDGNAParam.Append("</GroupID>");
//创建该临时组的DC用户号,此参数目前不使用，不用填写
strDGNAParam.Append("<DcID>");
strDGNAParam.Append("4120");
strDGNAParam.Append("</DcID>");
//该组别名，可以不填
strDGNAParam.Append("<Alias>");
strDGNAParam.Append("</Alias>");
//不用填写
strDGNAParam.Append("<Priority>");
strDGNAParam.Append("</Priority>");
//不用填写
strDGNAParam.Append("<MaxPeriod>");
strDGNAParam.Append("</MaxPeriod>");
//群组成员列表，最大值8个
strDGNAParam.Append("<GroupList>");
//群组成员ID
strDGNAParam.Append("<GroupID>");
strDGNAParam.Append("1001");
strDGNAParam.Append("</GroupID>");
strDGNAParam.Append("</GroupList>");
//用户成员列表，最大值200个
strDGNAParam.Append("<UserList>");
//用户成员ID
strDGNAParam.Append("<UserID>");
strDGNAParam.Append("4114");
strDGNAParam.Append("</UserID>");
strDGNAParam.Append("</UserList>");
strDGNAParam.Append("</Content>")
//调用创建临时群组接口
CString strResult = m_eLTE_Player.ELTE_OCX_CreateTempGroup(strDGNAParam);
//判定接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

注意

1. 临时群组没有删除接口，当无操作一段时间后会被自动回收。
2. 临时群组在组呼状态结束后会自动删除；如果不发起临时组呼业务这个群组不会自动删除，只有调度员退出或者进行临时群组业务时才会自动删除。

创建派接组

调用【**ELTE_OCX_CreatePatchGroup(创建派接组)**】接口，创建派接组入参包括创建者（调度机用户ID）、群组成员列表、派接组ID。

接口调用代码示例如下。

```
//cpp code
//创建一个派接组
//strPCHGRPParam为xml报文字符串
CString strPCHGRPParam;
strPCHGRPParam.Append("<Content>");
strPCHGRPParam.Append("<DcID>");
//创建派接组的调度员ID，此参数目前不使用，不用填写。
strPCHGRPParam.Append("4120");
strPCHGRPParam.Append("</DcID>");
//派接组的组号不需填写
strPCHGRPParam.Append("<PatchGroupID>");
strPCHGRPParam.Append("</PatchGroupID>");
//派接组名称
strPCHGRPParam.Append("<PatchGroupName>");
strPCHGRPParam.Append("test");
strPCHGRPParam.Append("</PatchGroupName>");
//派接组成员列表，最多支持20个普通组被派接
strPCHGRPParam.Append("<PatchGroupMemberList>");
//派接组成员，为普通群组
strPCHGRPParam.Append("<PatchGroupMember>");
strPCHGRPParam.Append("1001");
strPCHGRPParam.Append("</PatchGroupMember>");
strPCHGRPParam.Append("</PatchGroupMemberList>");
strPCHGRPParam.Append("</Content>");
//调用创建派接组接口
CString strResult =m_eLTE_Player.ELTE_OCX_CreatePatchGroup(strPCHGRPParam);
//判断调用派接组接口是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

若收到消息回调事件函数【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】中StatusType=24，StatusValue=4031，说明派接组创建成功。

增加派接组成员

调用【**ELTE_OCX_AddPatchGroupMember(增加派接组成员)**】接口，增加派接组成员的入参为创建者ID，派接组ID，派接组名称，增加成员列表。

接口调用代码示例如下。

```
//cpp code
//增加派接组成员
//构造strPCHGRPParam入参字符串
CString strPCHGRPParam;
strPCHGRPParam.Append("<Content>");
//派接组修改者，此参数目前不使用，不用填写。
strPCHGRPParam.Append("<DcID>");
strPCHGRPParam.Append("4120");
strPCHGRPParam.Append("</DcID>");
//派接组ID
strPCHGRPParam.Append("<PatchGroupID>");
strPCHGRPParam.Append("60001");
strPCHGRPParam.Append("</PatchGroupID>");
//派接组名称
strPCHGRPParam.Append("<PatchGroupName>");
strPCHGRPParam.Append("test");
strPCHGRPParam.Append("</PatchGroupName>");
strPCHGRPParam.Append("<PatchGroupMemberList>");
```

```
//增加派接组成员ID
strPCHGRPParam.Append("<PatchGroupMember>");
strPCHGRPParam.Append("1002");
strPCHGRPParam.Append("</PatchGroupMember>");
strPCHGRPParam.Append("</PatchGroupMemberList>");
strPCHGRPParam.Append("</Content>");
//调用增加派接组成员接口
CString strResult =m_eLTE_Player.ELTE_OCX_AddPatchGroupMember (strPCHGRPParam);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

若收到消息回调事件函数【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】中StatusType=24，StatusValue=4035，说明派接组增加成员成功。

删除派接组成员

调用【**ELTE_OCX_DeletePatchGroupMember(删除派接组成员)**】接口，调度员删除自己创建的派接组内成员，只有该派接组的创建者才可以删除派接组成员。

接口调用代码示例如下。

```
//cpp code
//删除派接组成员
//定义并构造strPCHGRPParam入参字符串
CString strPCHGRPParam;
strPCHGRPParam.Append("<Content>");
//派接组的创建者，此参数目前不使用，不用填写。
strPCHGRPParam.Append("<DcID>");
strPCHGRPParam.Append("4120");
strPCHGRPParam.Append("</DcID>");
//派接组ID
strPCHGRPParam.Append("<PatchGroupID>");
strPCHGRPParam.Append("60001");
strPCHGRPParam.Append("</PatchGroupID>");
//派接组名称
strPCHGRPParam.Append("<PatchGroupName>");
strPCHGRPParam.Append("test");
strPCHGRPParam.Append("</PatchGroupName>");
strPCHGRPParam.Append("<PatchGroupMemberList>");
//待删除的派接组成员
strPCHGRPParam.Append("<PatchGroupMember>");
strPCHGRPParam.Append("1002");
strPCHGRPParam.Append("</PatchGroupMember>");
strPCHGRPParam.Append("</PatchGroupMemberList>");
strPCHGRPParam.Append("</Content>");
CString strResult =m_eLTE_Player.ELTE_OCX_DeletePatchGroupMember (strPCHGRPParam);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

若收到消息回调事件函数【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】中StatusType=24，StatusValue=4037，说明派接组删除成员成功

删除派接组

调用【**ELTE_OCX_CancelPatchGroup(删除派接组)**】接口，删除派接组的入参为派接组ID。

接口调用代码示例如下。

```
//cpp code
//删除派接组99899997
```

```
CString strResult =m_eLTE_Player.ELTE_OCX_CancelPatchGroup ("99899997");
//判定接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

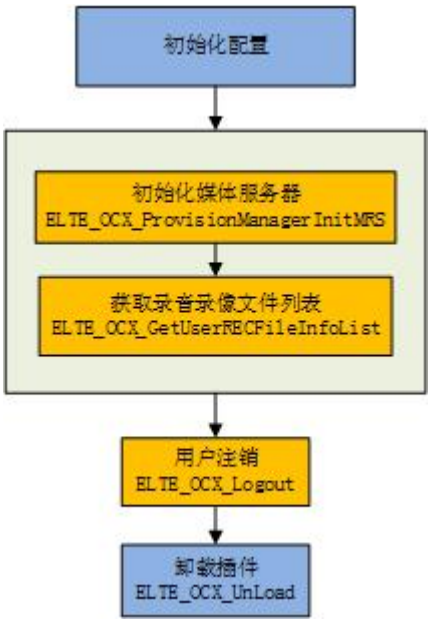
若收到消息回调事件函数【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】中StatusType=24，StatusValue=4033，说明删除派接组成功。

6.1.3 获取录音录像文件列表

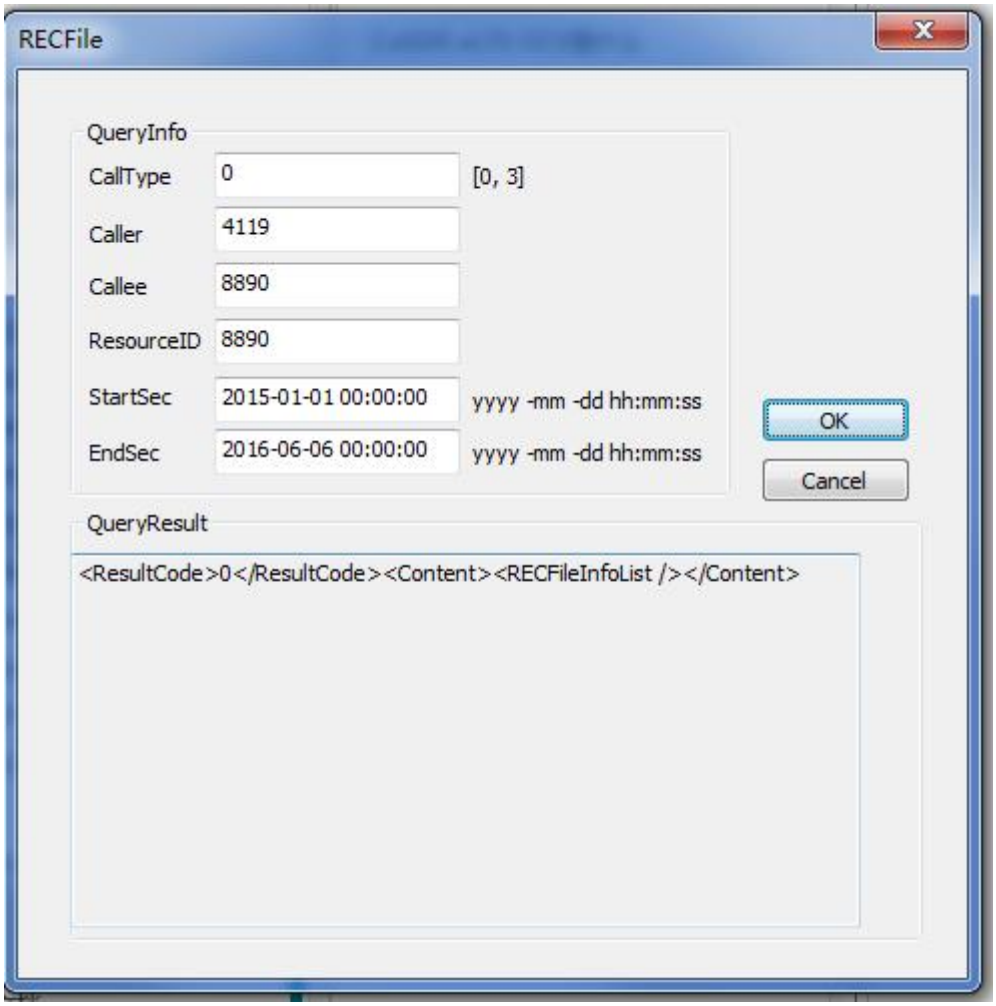
本章节的开发任务是介绍怎么获取调用录音录像文件列表，包括调用流程、调用方法、注意事项等。

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界面：



前提条件

1. 已完成**初始化配置**。
2. 调度台有查询录音录像文件的权限。

初始化媒体服务器

成功登录调度机，并接收到自动下载配置数据通知事件之后调用**【ELTE_OCX_ProvisionManagerInitMRS(初始化媒体服务器)】**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。

接口调用代码示例如下。

```
//cpp code
//初始化媒体服务器，媒体服务器IP：172.22.9.105
CString iResult = m_peLTE_Player.ELTE_OCX_ProvisionManagerInitMRS("172.22.9.105");
if(ELTE_OCX_ERR_SUCCESS == iResult)
{
    //接口调用成功
}
```

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。其他错误码参考**【eLTE产品错误码】**。

获取录音录像文件列表

调用【**ELTE_OCX_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

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。

接口调用代码示例如下。

```
//cpp code
//构造一个查询条件字符串，先定义查询条件变量
CString strQuery;
CString m_CallType = "2";
CString m_Caller = "-1";
CString m_Callee = "-1";
CString m_ResourceID = "8895";
CString m_StartSec = "";
CString m_EndSec = "";
strQuery.Append(_T("<Content>"));
strQuery.Append(_T("<RECQueryInfo>"));
//CallType是呼叫类型，0语音点呼，1视频点呼，2视频回传，3群组呼叫
strQuery.Append(_T("<CallType>"));
strQuery.Append(m_CallType);
strQuery.Append(_T("</CallType>"));
//Caller是主叫号码，默认值可填-1
strQuery.Append(_T("<Caller>"));
strQuery.Append(m_Caller);
strQuery.Append(_T("</Caller>"));
//Callee是被叫号码，默认值可填-1
strQuery.Append(_T("<Callee>"));
strQuery.Append(m_Callee);
strQuery.Append(_T("</Callee>"));
//ResourceID是资源ID（包括群组ID和终端ID），默认值可填-1
strQuery.Append(_T("<ResourceID>"));
strQuery.Append(m_ResourceID);
strQuery.Append(_T("</ResourceID>"));
//查询开始时间，格式如：yyyy-mm-dd hh:mm:ss
strQuery.Append(_T("<StartSec>"));
strQuery.Append(m_StartSec);
strQuery.Append(_T("</StartSec>"));
//查询结束时间，格式如：yyyy-mm-dd hh:mm:ss
strQuery.Append(_T("<EndSec>"));
strQuery.Append(m_EndSec);
strQuery.Append(_T("</EndSec>"));
strQuery.Append(_T("</RECQueryInfo>"));
strQuery.Append(_T("</Content>"));
//调用查询录音录像文件列表接口
CString strResult = m_eLTE_Player.ELTE_OCX_GetUserRECFileInfoList(strQuery);
```

接口调用成功后strResult返回结果码和查询录音录像文件信息，如下。

```
//xml code
<Content>
    <ResultCode>0</ResultCode>
    <RECFileInfoList>
        <RECFileInfo>
            <CallType>0</CallType>
            <Caller>8895</Caller>
            <Callee>4116</Callee>
            <ResourceID>0</ResourceID>
```

```
            <StartSec>2016-09-13 21:45:19</StartSec>
            <EndSec>2016-09-13 21:45:20</EndSec>
            <UrlFTP>http://172.22.9.120:8000/ubp/rec/
2016/09/13/21/8895_4116_20160913214517/8895_4116_20160913214519_584066.mp4</UrlFTP>
            <UrlRTSP>rtsp://172.22.9.120:8554/1_4e08620a-79b8-11e6-8000-4cb16cf61a04_1</UrlRTSP>
        </RECFileInfo>
        <RECFileInfo>
            <CallType>0</CallType>
            <Caller>8895</Caller>
            <Callee>4114</Callee>
            <ResourceID>0</ResourceID>
            <StartSec>2016-09-13 20:47:15</StartSec>
            <EndSec>2016-09-13 20:47:16</EndSec>
            <UrlFTP>http://172.22.9.120:8000/ubp/rec/
2016/09/13/20/8895_4114_20160913204712/8895_4114_20160913204715_104348.mp4</UrlFTP>
            <UrlRTSP>rtsp://172.22.9.120:8554/1_311d3e70-79b0-11e6-8000-4cb16cf61a04_1</UrlRTSP>
        </RECFileInfo>
    </RECFileInfoList>
</Content>
```

UrlFTP可以用浏览器打开用调度台用户鉴权后回放录音录像文件，UrlRTSP需要使用VLC mediaplayer工具打开，鉴权后可回放录音录像文件。

6.2 语音调度

6.2.1 概述

华为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。

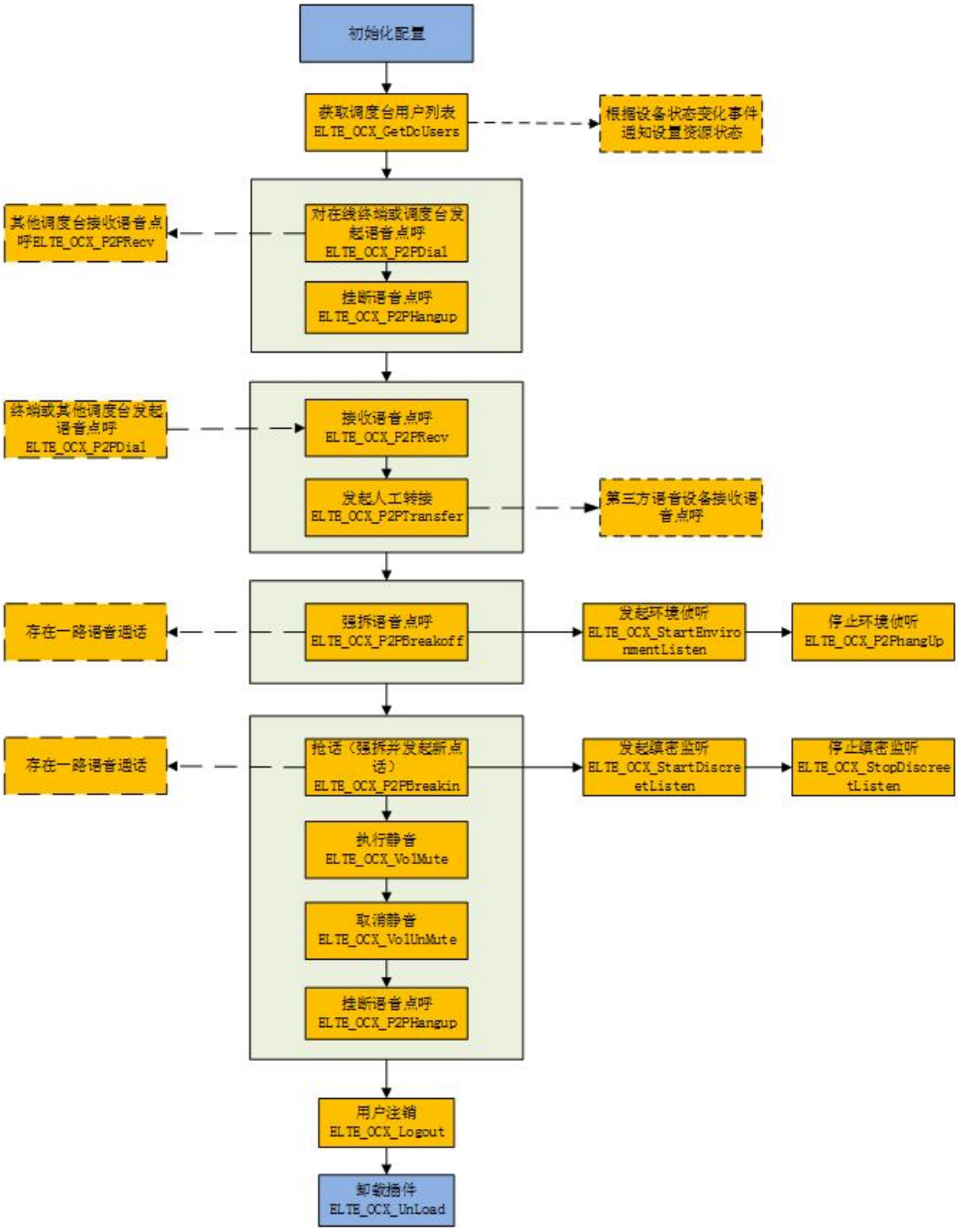
我们将按照如下场景进行详细的分析和介绍：

- 语音点呼场景
- 语音组呼场景

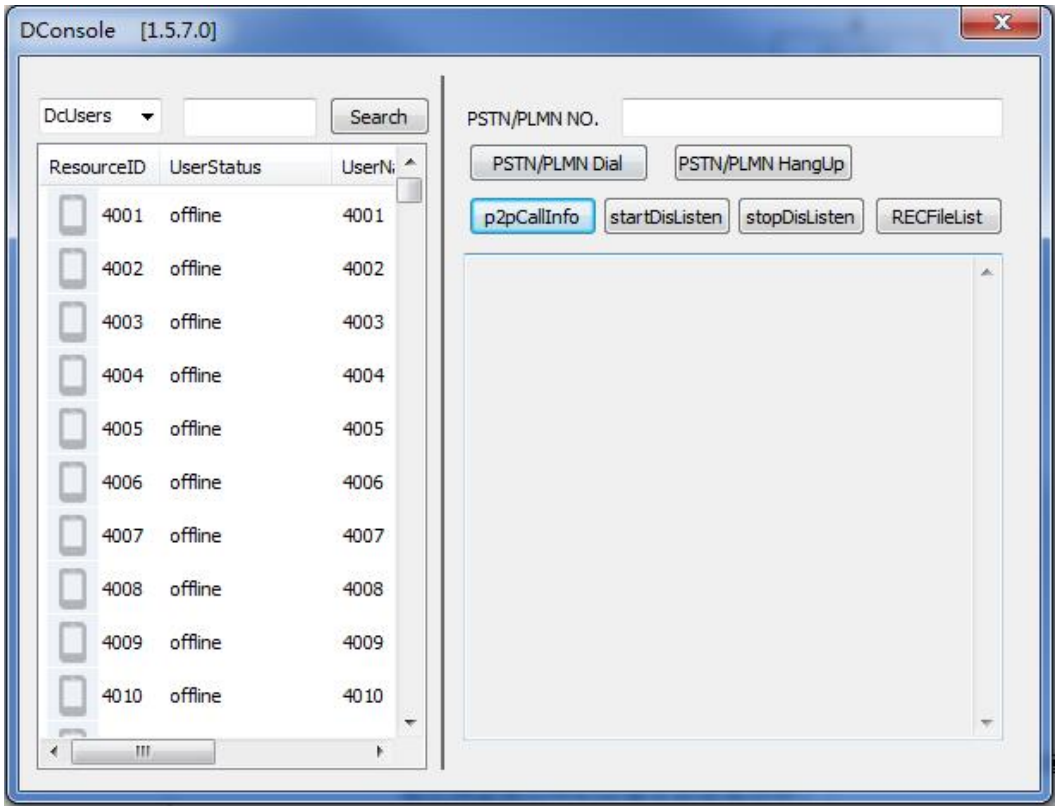
6.2.2 语音点呼场景

本章节的开发任务：语音点呼相关接口的调用方法和流程，演示特性开发的过程。

语音点呼接口调用流程：



语音调度Sample界面：

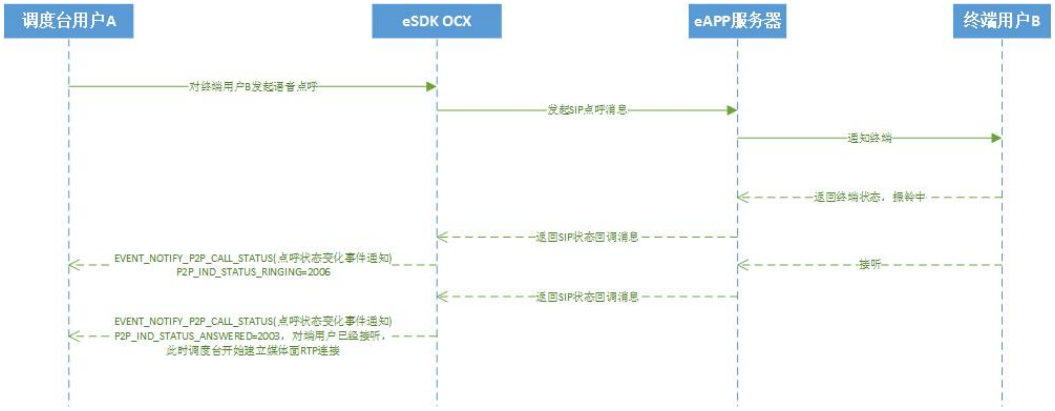


前提条件

已完成**初始化配置**，初始化配置中必须设置使用媒体面。

发起语音点呼

调用【**ELTE_OCX_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。



接口调用代码示例如下。

```
//cpp code
//对一个集群终端8890发起语音点呼
CString strResult = m_peLTE_Player.ELTE_OCX_P2PDial("8890");
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
```

```
    //接口调用成功
}
```

接口调动成功后，语音点呼状态需通过消息回调事件函数【**ELTE_OCX_Event(消息回调事件函数)**】中【**EVENT_NOTIFY_P2P_CALL_STATUS(点呼状态变化事件通知)**】来解析出点呼状态。消息回调事件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Callee、抢权者Inserter、被抢权者Targeter、音频格式Soundtype等。消息回调事件函数返回消息格式如下。

```
//xml code
<Content>
    <P2pcallStatus>2006</P2pcallStatus>
    <Caller>4119</Caller>
    <Callee>8890</Callee>
    <Inserter>0</Inserter>
    <Targeter>0</Targeter
    <Transfer />
    <LocalPort>0</LocalPort>
    <RemotePort>0</RemotePort>
    <RemoteIP>0.0.0.0</RemoteIP>
    <SoundPtype>-1</SoundPtype>
    <CallID>0</CallID>
    <SignalError>0</SignalError>
    <FromString>4119</FromString>
    <ToString>8890</ToString>
</Content>
```

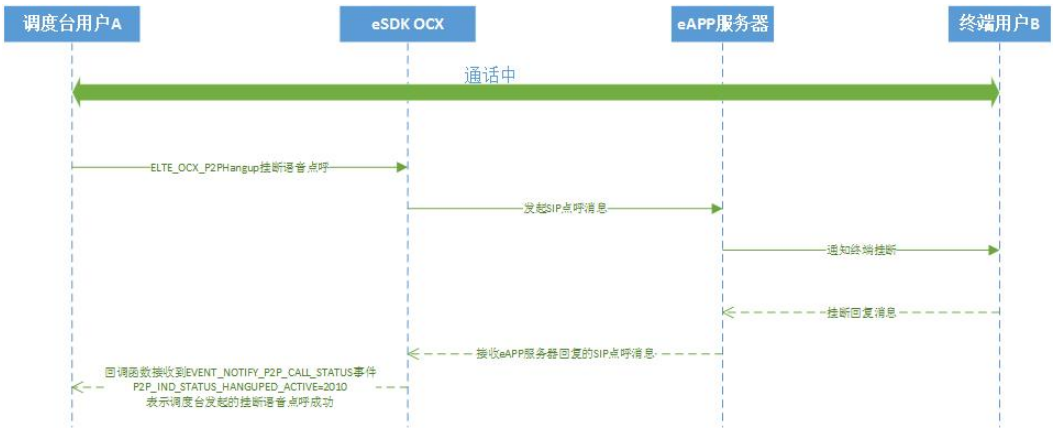
其中P2pcallStatus代表状态事件，有如下状态：

- 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【**EVENT_NOTIFY_P2P_CALL_STATUS(点呼状态变化事件通知)**】。

挂断语音点呼

调用【**ELTE_OCX_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。



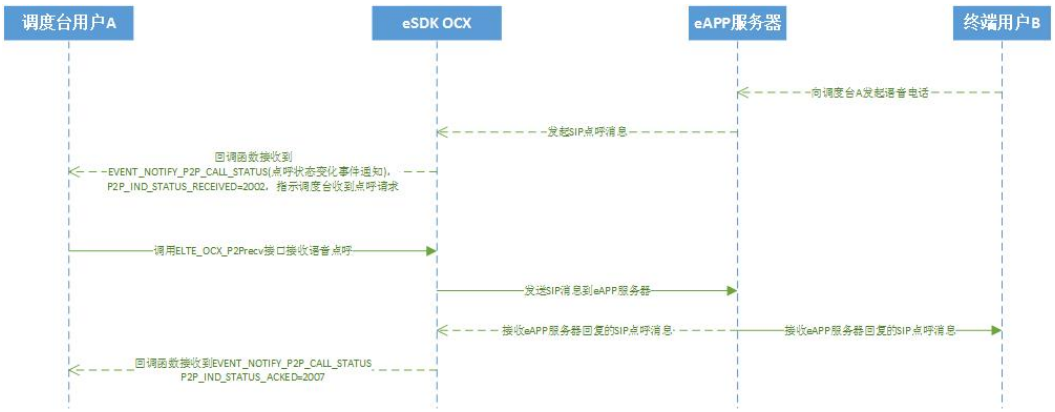
接口调用代码示例如下。

```
//cpp code
//挂断调度台与终端8890的语音点呼
CString strResult = m_peLTE_Player.ELTE_OCX_P2PHangup("8890");
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //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【EVENT_NOTIFY_P2P_CALL_STATUS(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，首先是解析消息回调事件函数中的点呼状态事件，如果点呼状态事件P2pcallStatus=2002，则调度台可以选择调用【ELTE_OCX_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。



调度台接收到终端语音电话的消息格式如下。

```
//xml code
<Content>
    <P2pcallStatus>2002</P2pcallStatus>
    <Caller>8890</Caller>
    <Callee>4116</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>8890</FromString>
    <ToString>4116</ToString>
</Content>
```

接口调用代码示例如下。

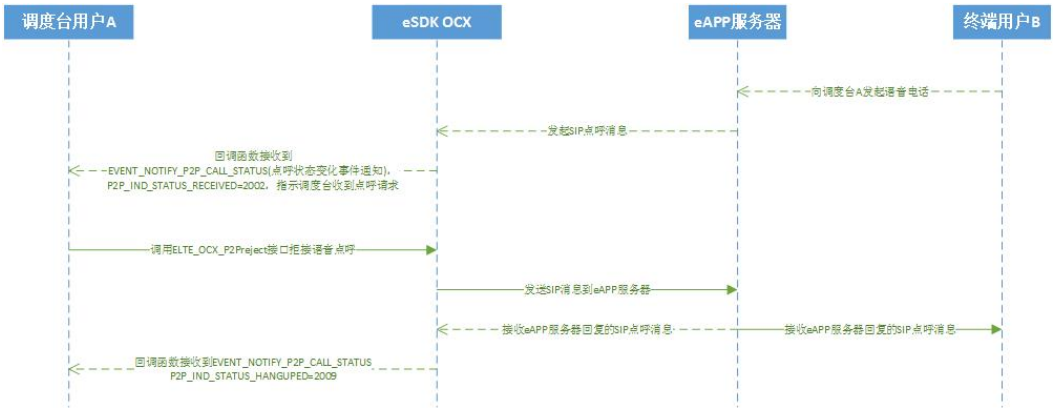
```
//cpp code
//接听来自终端8890的语音点呼
CString strResult = m_peLTE_Player.ELTE_OCX_P2PRecv("8890");
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

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【EVENT_NOTIFY_P2P_CALL_STATUS(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，首先是解析消息回调事件函数中的点呼状态事件，如果点呼状态事件p2pcallStatus=2002，则调度台可以选择调用

【ELTE_OCX_P2Preject(拒接语音点呼)】接口来拒接，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如下。

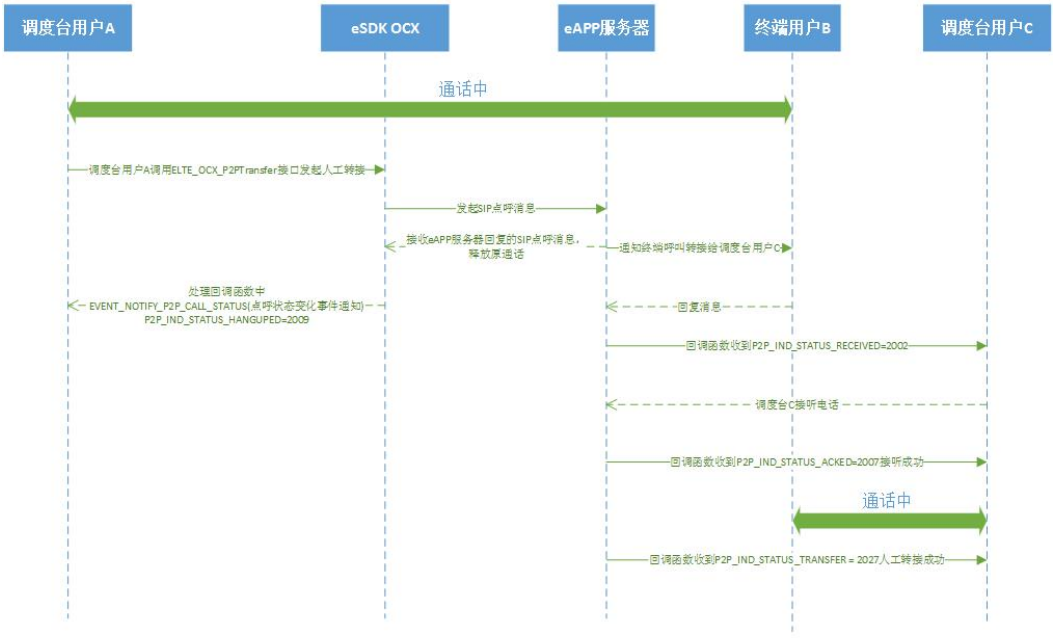
```
//cpp code
//拒接来自终端8890的语音点呼
CString strRst = m_peLTE_Player.ELTE_OCX_P2PReject("8890");
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功，收到状态事件P2pcallStatus=2009表示拒绝语音点呼成功。

其他状态请参考【EVENT_NOTIFY_P2P_CALL_STATUS(点呼状态变化事件通知)】。

发起人工转接

调用【ELTE_OCX_P2PTransfer(发起人工转接)】接口发起人工转接，使得本调度台与其他用户正在进行的语音通话转接给其他的用户，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。



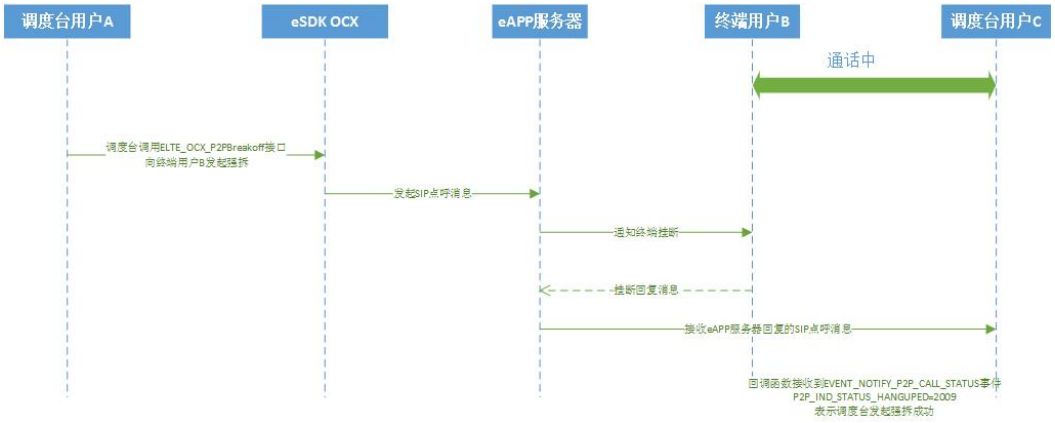
接口调用示代码示例如下。

```
//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("<ObjectID>");
strP2PTransferParam.Append("4135");
strP2PTransferParam.Append("</ObjectID>");
strP2PTransferParam.Append("</Content>");
//调用人工转接接口
CString strResult =m_eLTE_Player.ELTE_OCX_P2PTransfer("4116",strP2PTransferParam);
//判断人工转接接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

发起人工转接的调度台收到点呼状态事件为P2P_IND_STATUS_HANGUPED=2009语音挂断，人工转接目的调度台收到点呼状态事件P2pcallStatus=2027表示人工转接成功，其他状态请参考**EVENT_NOTIFY_P2P_CALL_STATUS(点呼状态变化事件通知)**。

强拆语音点呼

调用【**ELTE_OCX_P2Pbreakoff(强拆点呼)**】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。



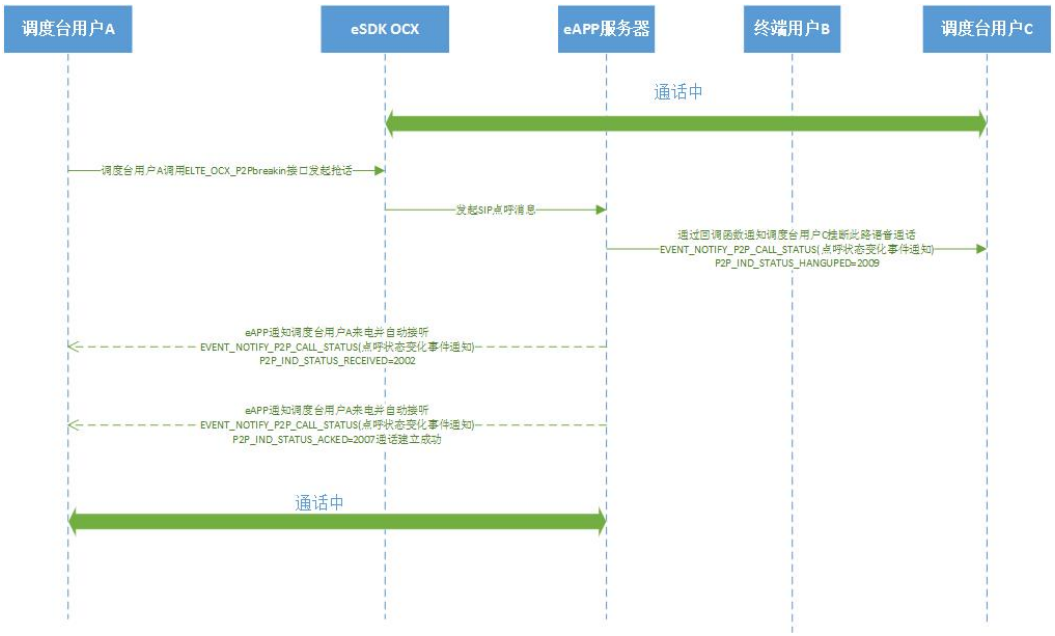
接口调用示代码示例如下。

```
//cpp+ code
//调度台强拆终端8895与其他用户的通话
CString strResult = m_eLTE_Player.ELTE_OCX_P2PBreakoff("8895");
//判断强拆接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0接口调用成功，其他错误码可参考【**eLTE产品错误码**】。

抢话语音点呼

调用【**ELTE_OCX_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。



接口调用示代码示例如下。

```
//cpp code
//调度台对手持终端8895发起抢话
CString strResult = m_eLTE_Player.ELTE_OCX_P2PBreakin("8895");
//判断抢话接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，其他错误码参考【**eLTE产品错误码**】。

执行静音

调用【**ELTE_OCX_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
CString strResult = m_eLTE_Player.ELTE_OCX_VolMute("8895",strMuteParam);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
```

```
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，其他错误码参考【**eLTE产品错误码**】。

取消静音

调用【**ELTE_OCX_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
CString strResult = m_eLTE_Player.ELTE_OCX_VolUnMute("8895",strMuteParam);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，其他错误码参考【**eLTE产品错误码**】。

发起环境侦听

调用【**ELTE_OCX_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。

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
CString strResult =m_eLTE_Player.ELTE_OCX_StartEnvironmentListen("8895");
//判断环境侦听接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，其他错误码参考【**eLTE产品错误码**】。

结束环境侦听

结束环境侦听与【**ELTE_OCX_P2Phangup(挂断语音点呼)**】接口相同。

发起缜密监听

调用【**ELTE_OCX_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。

接口调用示代码示例如下。

```
//cpp code
//调度台对手持终端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
CString strResult = m_eLTE_Player.ELTE_OCX_StartDiscreetListen ("8890");
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，其他错误码参考【**eLTE产品错误码**】。

停止缜密监听

调用【**ELTE_OCX_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。

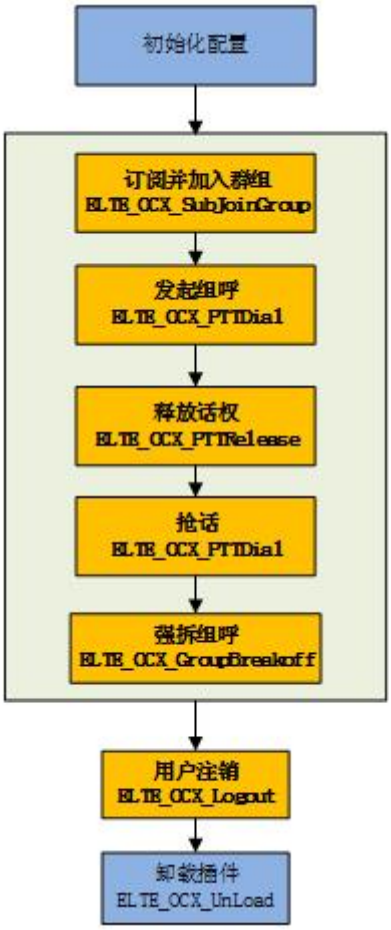
```
//cpp code
//调度台4119对手持终端8890停止缜密监听,调度台停止缜密监听的对象可以是调度台用户ID、群组ID或手持终端ID
CString strResult = m_eLTE_Player.ELTE_OCX_StopDiscreetListen ("1001");
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，其他错误码参考【**eLTE产品错误码**】。

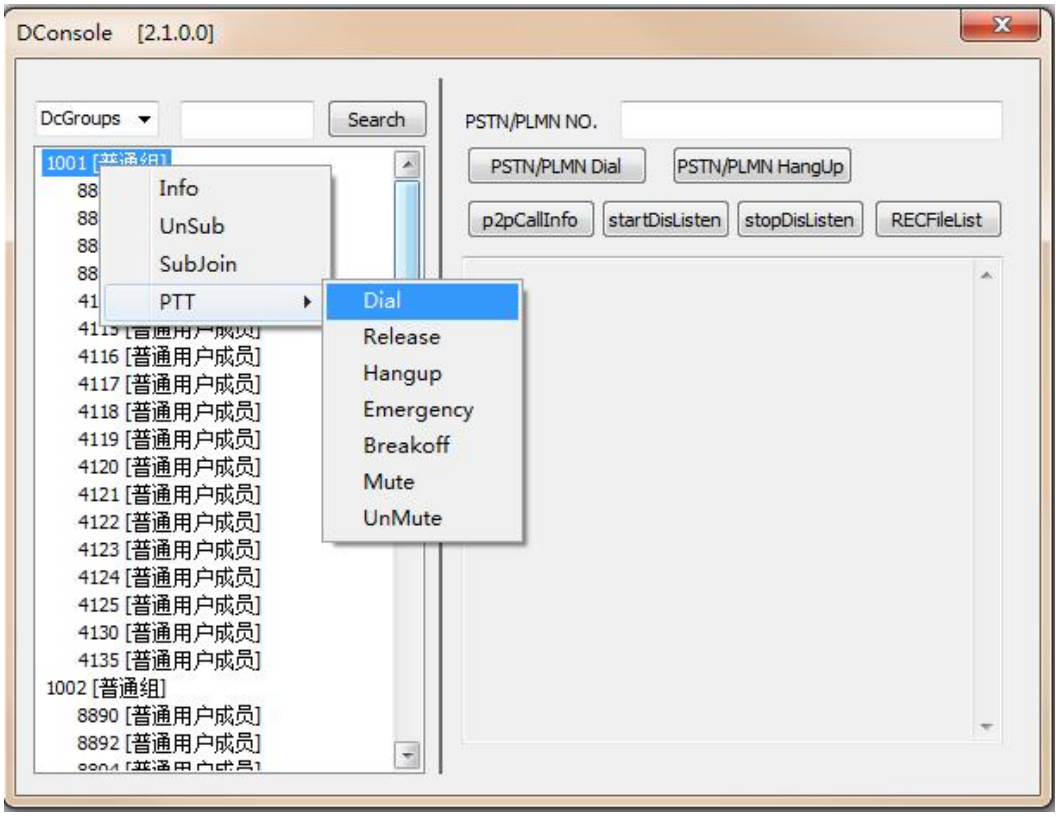
6.2.3 语音组呼场景

本章节的开发任务：开发语音组呼的相关接口调用的方法和流程，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

已完成**初始化配置**，初始化配置中必须设置使用媒体面。

订阅并加入群组

调用【**ELTE_OCX_SubJoinGroup(订阅并自动加入群组)**】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。可接收群和发起群组组呼前提为该调度台用户在该群组中。

接口调用代码示例如下。

```
//cpp code
//订阅并加入1001群组
CString strResult = m_eLTE_Player.ELTE_OCX_SubJoinGroup("1001");
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功后，解析消息回调事件函数中【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】，消息回调事件函数返回的消息格式如下。

```
//xml code
<Content>
    <ResourceID>1001</ResourceID>
    <ResourceName></ResourceName>
```

```
    <StatusType>11</StatusType>
    <StatusValue>4003</StatusValue>
    <AttachingGroup>0</AttachingGroup>
</Content>
```

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

取消订阅群组

调用【**ELTE_OCX_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
CString strResult = m_eLTE_Player.ELTE_OCX_UnSubscribeGroup("1001");
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功后，解析消息回调事件函数中【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】，消息回调事件函数返回的消息格式如下。

```
//xml code
<Content>
    <ResourceID>1001</ResourceID>
    <ResourceName></ResourceName>
    <StatusType>11</StatusType>
    <StatusValue>4004</StatusValue>
    <AttachingGroup>0</AttachingGroup>
</Content>
```

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

发起组呼或抢话

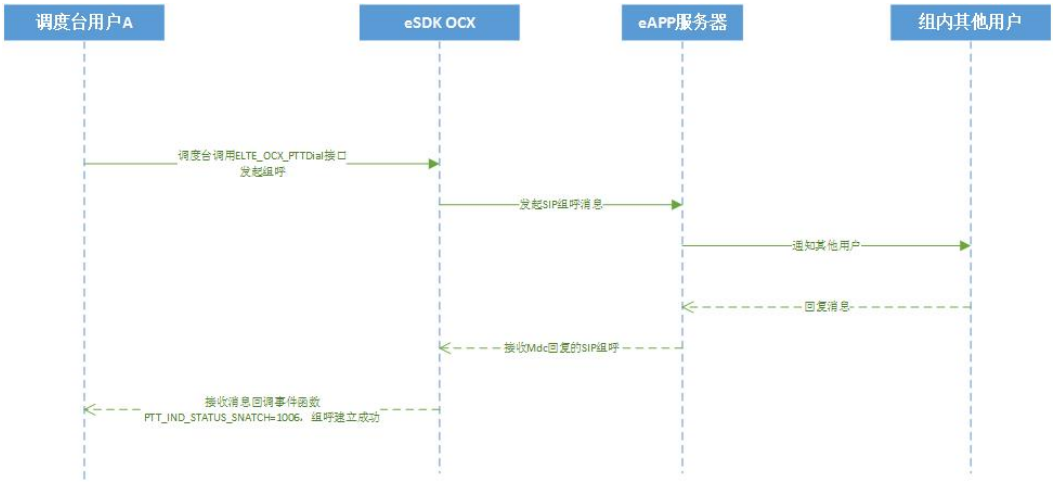
【**ELTE_OCX_PTTDial(发起语音点呼)**】接口用于发起组呼（固定、临时和动态组的组呼）或抢权，即调度台发起群组通话或在群组通话中抢权。若该群组当前已有活动通话，则调度台申请话权。

注意

1.如果调度台用户不属于这个群组调用【**ELTE_OCX_SubJoinGroup(订阅并自动加入群组)**】接口后，可以接收组呼但不能对该群组发起组呼。

2.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
CString strResult = m_eLTE_Player.ELTE_OCX_PTTDial("1001");
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，解析消息回调事件函数中事件类型iEventType =6的组呼状态变化事件通知事件，返回消息格式如下。

```
//xml code
<Content>
    <GroupID>1001</GroupID>
    <GroupCallStatus>1006</GroupCallStatus>
    <Speaker>4116</Speaker>
    <LocalPort>64082</LocalPort>
    <RemotePort>28034</RemotePort>
    <RemoteIP>120.9.22.172</RemoteIP>
    <SoundPtype>-1</SoundPtype>
    <SpeakerName>4116</SpeakerName>
    <CallID>0</CallID>
    <SignalError>-1</SignalError>
    <FromString></FromString>
    <ToString></ToString>
</Content>
```

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【**EVENT_NOTIFY_GROUP_STATUS(组呼状态变化事件通知)**】。

释放组呼话权

【**ELTE_OCX_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。

接口调用代码示例如下。

```
//cpp code
//释放1001群组话权
CString strResult = m_eLTE_Player.ELTE_OCX_PTTRelease("1001");
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，解析消息回调事件函数中事件类型iEventType =6的组呼状态变化事件通知事件，返回消息格式如下。

```
//xml code
<Content>
    <GroupID>1001</GroupID>
    <GroupCallStatus>1012</GroupCallStatus>
    <Speaker>0</Speaker>
    <LocalPort>0</LocalPort>
    <RemotePort>0</RemotePort>
    <RemoteIP>0.0.0.0</RemoteIP>
    <SoundPtype>-1</SoundPtype>
    <SpeakerName></SpeakerName>
    <CallID>0</CallID>
    <SignalError>-1</SignalError>
    <FromString></FromString>
    <ToString></ToString>
</Content>
```

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

紧急组呼

【**ELTE_OCX_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
CString strResult =m_eLTE_Player.ELTE_OCX_PTTEmergency ("1001");
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，解析消息回调事件函数中事件类型iEventType =6的组呼状态变化事件通知事件，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></FromString>
    <ToString></ToString>
</Content>
```

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

强拆组呼

【**ELTE_OCX_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。

接口调用示代码示例如下。

```
//cpp code
//对群组1001发起强拆组呼
CString strResult =m_eLTE_Player.ELTE_OCX_GroupBreakoff ("1001");
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，解析消息回调事件函数中事件类型iEventType =6的组呼状态变化事件通知事件，返回消息格式如下。

```
//xml code
<Content>
    <GroupID>1001</GroupID>
    <GroupCallStatus>1008</GroupCallStatus>
    <Speaker>0</Speaker>
    <LocalPort>0</LocalPort>
    <RemotePort>0</RemotePort>
    <RemoteIP>0.0.0.0</RemoteIP>
    <SoundPtype>-1</SoundPtype>
    <SpeakerName></SpeakerName>
    <CallID>0</CallID>
    <SignalError>-1</SignalError>
    <FromString></FromString>
    <ToString></ToString>
</Content>
```

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

发起群组缜密监听

参考**语音点呼场景发起缜密监听**，将入参变更为群组ID。

停止群组缜密监听

参考**语音点呼场景停止缜密监听**，将入参变更为群组ID。

6.3 视频调度

6.3.1 概述

华为eSDK eLTE SDK开放视频调度接口，视频调度接口包括发起视频回传、接收视频回传或视频分发、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。

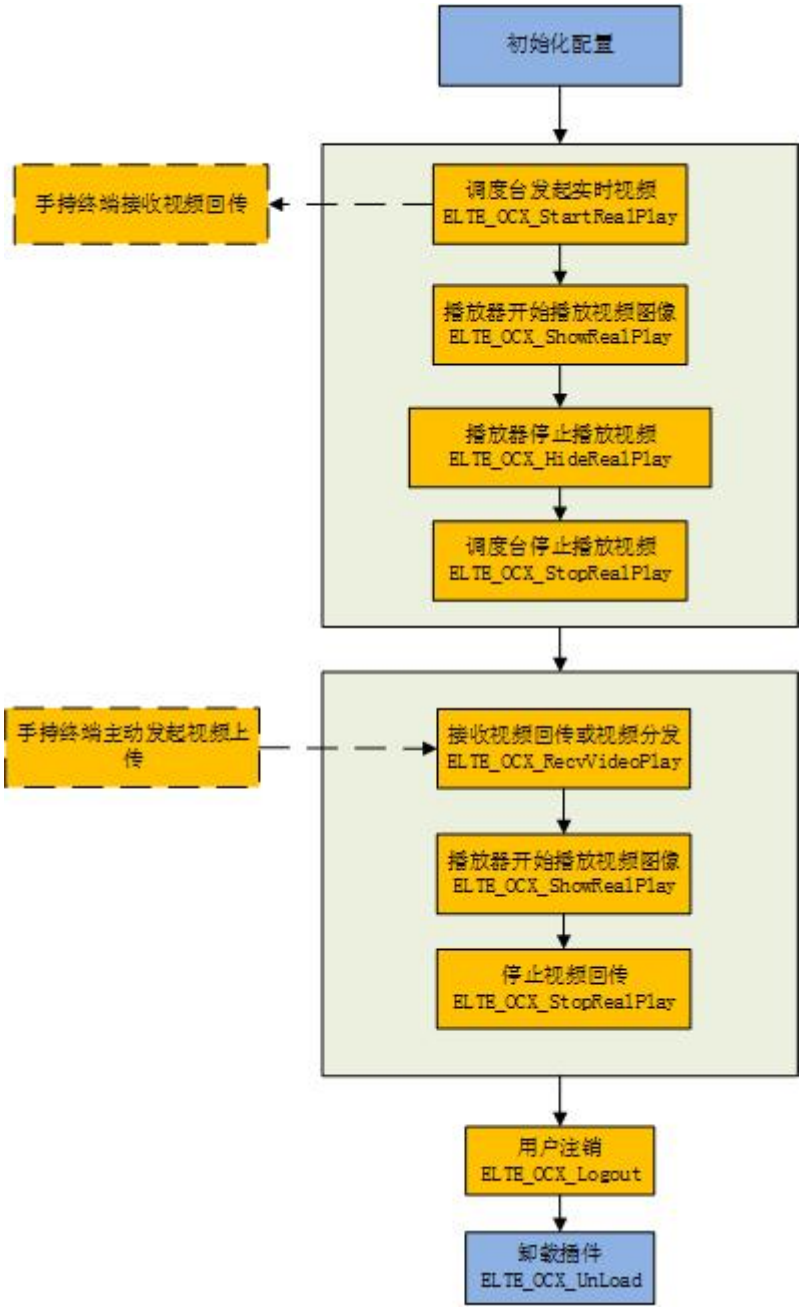
我们将按照下面的三种场景来分别介绍这些接口。

- **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

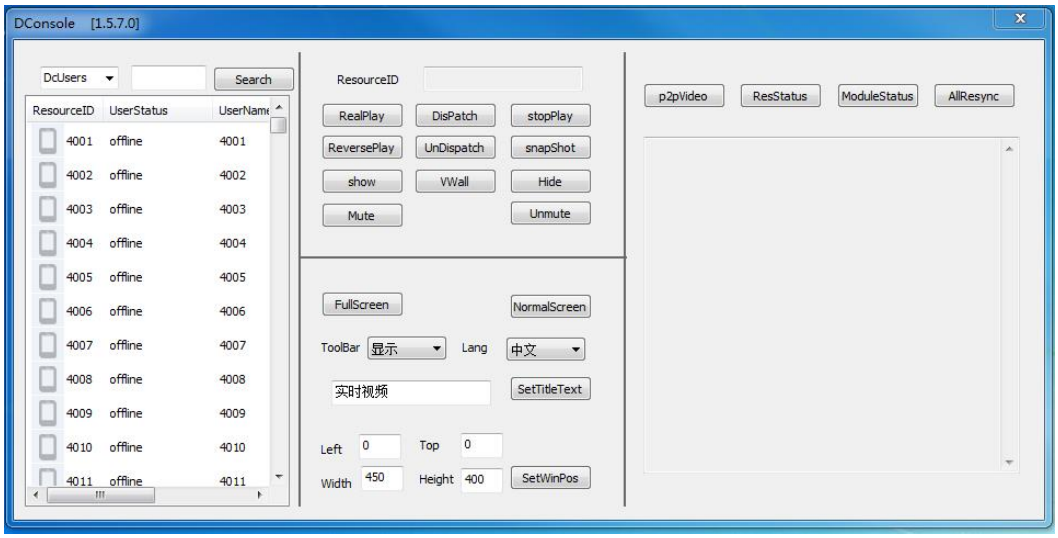
本章节将详细介绍视频回传场景下接口调用的流程方法和注意事项。

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，和手持终端主动发起视频上传两种方式，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

已完成**初始化配置**，初始化配置中必须设置使用媒体面。

调度台发起实时视频

调用【**ELTE_OCX_StartRealPlay(调度台发起实时视频)**】接口向手持终端或IVS摄像头发起实时视频回传，调用接口前需要准备视频参数XML，其中：

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。

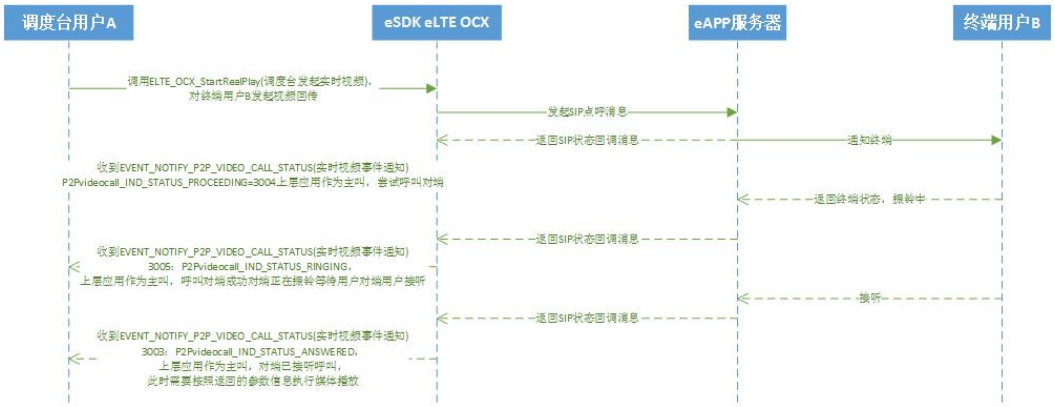
MuteType表示是否需要伴音，0代表需要伴音，1代表无伴音。

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。详细需要解析消息回调事件函数**EVENT_NOTIFY_P2P_VIDEO_CALL_STATUS(实时视频事件通知)**，CallStatus=3003情况下VideoFormatType参数取值。

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    1：无伴音
strVideoParam.Append("<MuteType>");
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
CString strResult = m_eLTE_Player.ELTE_OCX_StartRealPlay("8894",strVideoParam);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，实际视频回传发起是否成功需要解析消息回调事件函数【**EVENT_NOTIFY_P2P_VIDEO_CALL_STATUS(实时视频事件通知)**】，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_OCX_ShowRealPlay(播放器开始播放视频图像)**】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【**EVENT_NOTIFY_P2P_VIDEO_CALL_STATUS(实时视频事件通知)**】。

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>
    <Camera>0</Camera>
    <SoundPtype>0</SoundPtype>
    <VideoFormatType>5</VideoFormatType>
    <CallID>0</CallID>
    <SignalError>-1</SignalError>
    <FromString>4116</FromString>
    <ToString>8894</ToString>
</Content>
```

播放器开始播放视频图像

调度台发起向手持终端或IVS摄像头发起视频回传成功后，需要调用【**ELTE_OCX_ShowRealPlay(播放器开始播放视频图像)**】来开始播放视频图像。

说明

1. 在调用【**ELTE_OCX_StartRealPlay(调度台发起实时视频)**】接口之后调用此接口。
2. 调用此接口前需要先提取【**EVENT_NOTIFY_P2P_VIDEO_CALL_STATUS(实时视频事件通知)**】中CallStatus=3003时的本地音频端口、本地视频端口、服务器音频端口、服务器视频端口等信息。

先定义一个提取xml字符串中字符的函数，如下。

```
//cpp code
CString XmlParseElemValue(const CString& xmlStr, const CString& elem)
播放{
    CString value(xmlStr);
    CString elemBegin(elem);
    CString elemEnd(elem);
    elemBegin.Insert(0, _T("<"));
    elemBegin.Append(_T(">"));
    elemEnd.Insert(0, _T("</"));
    elemEnd.Append(_T(">"));
    value = value.Left(value.Find(elemEnd));
    value = value.Right(value.GetLength() - elemBegin.GetLength() - value.Find(elemBegin));
    return value;
}
```

然后调用开始提取和构造播放视频图像的入参xml字符串以及接口调用，代码示例如下。

```
//cpp code
//提取本地音频端口号
CString strLocalAudioPort = XmlParseElemValue(str, "LocalAudioPort");
//提取本地视频端口号
CString strLocalVideoPort = XmlParseElemValue(str, "LocalVideoPort");
//提取远端音频端口号
CString strRemoteAudioPort = XmlParseElemValue(str, "RemoteAudioPort");
//提取远端视频端口号
CString strRemoteVideoPort = XmlParseElemValue(str, "RemoteVideoPort");
//提取视频源ID
CString strVideoResourceID = XmlParseElemValue(str, "Uri");
//构造本地流媒体地址参数
CString strLocalMediaAddr;
strLocalMediaAddr.Append("<Content>");
strLocalMediaAddr.Append("<LocalMediaAddr>");
```

```
//本地IP
strLocalMediaAddr.Append("<LocalIP>");
strLocalMediaAddr.Append(m_strLocalIP);
strLocalMediaAddr.Append("</LocalIP>");
//本地视频端口
strLocalMediaAddr.Append("<VideoPort>");
strLocalMediaAddr.Append(strLocalVideoPort);
strLocalMediaAddr.Append("</VideoPort>");
//本地音频端口
strLocalMediaAddr.Append("<AudioPort>");
strLocalMediaAddr.Append(strLocalAudioPort);
strLocalMediaAddr.Append("</AudioPort>");
strLocalMediaAddr.Append("</LocalMediaAddr>");
strLocalMediaAddr.Append("</Content>");
//构造远端流媒体地址参数
CString strRemoteMediaAddr;
strRemoteMediaAddr.Append("<Content>");
strRemoteMediaAddr.Append("<RemoteMediaAddr>");
//服务器IP
strRemoteMediaAddr.Append("<RemoteIP>");
strRemoteMediaAddr.Append(m_strServerIP);
strRemoteMediaAddr.Append("</RemoteIP>");
//服务器视频端口
strRemoteMediaAddr.Append("<VideoPort>");
strRemoteMediaAddr.Append(strRemoteVideoPort);
strRemoteMediaAddr.Append("</VideoPort>");、
//服务器音频端口
strRemoteMediaAddr.Append("<AudioPort>");
strRemoteMediaAddr.Append(strRemoteAudioPort);
strRemoteMediaAddr.Append("</AudioPort>");
strRemoteMediaAddr.Append("</RemoteMediaAddr>");
strRemoteMediaAddr.Append("</Content>");

//调用ELTE_OCX_ShowRealPlay(播放器开始播放视频图像)接口
CString strResult =
m_eLTE_Player.ELTE_OCX_ShowRealPlay(strVideoResourceID ,strLocalMediaAddr,strRemoteMediaAddr);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，其他请参考【**错误码**】。

播放器停止播放视频

调用【**ELTE_OCX_HideRealPlay(播放器停止播放视频)**】接口停止播放视频画面。

说明

1. 调用该接口后，只是关闭的媒体的显示，还需要调用**调度台停止播放视频**关闭视频流。
2. 如果不调用**调度台停止播放视频**关闭实时视频，可以通过**播放器开始播放视频图像**继续播放视频。

接口调用代码示例如下。

```
//cpp code
//播放器停止视频播放
CString strResult = m_eLTE_Player.ELTE_OCX_HideRealPlay();
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功返回ResultCode为0，其他请参考【**错误码**】。

调度台停止播放视频

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_OCX_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8894的视频回传
CString strResult = m_eLTE_Player.ELTE_OCX_StopRealPlay("8894");
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

解析消息回调事件函数【EVENT_NOTIFY_P2P_VIDEO_CALL_STATUS(实时视频事件通知)】，返回的消息格式如下。

```
//xml code
<Content>
    <CallStatus>3009</CallStatus>
    <Callee>8894</Callee>
    <Caller>4116</Caller>
    <LocalAudioPort>0</LocalAudioPort>
    <LocalVideoPort>0</LocalVideoPort>
    <RemoteAudioPort>0</RemoteAudioPort>
    <RemoteVideoPort>0</RemoteVideoPort>
    <RemoteIp>0.0.0.0</RemoteIp>
    <Uri>8894</Uri>
    <Channel>65535</Channel>
    <SoundMute>0</SoundMute>
    <UserConfirm>-1</UserConfirm>
    <Camera>-1</Camera>
    <SoundPtype>-1</SoundPtype>
    <VideoFormatType>5</VideoFormatType>
    <CallID>4294967295</CallID>
    <SignalError>-1</SignalError>
    <FromString></FromString>
    <ToString></ToString>
</Content>
```

当CallStatus=3009表示调度台主动停止视频回传成功，其他CallStatus值请参考【EVENT_NOTIFY_P2P_VIDEO_CALL_STATUS(实时视频事件通知)】。

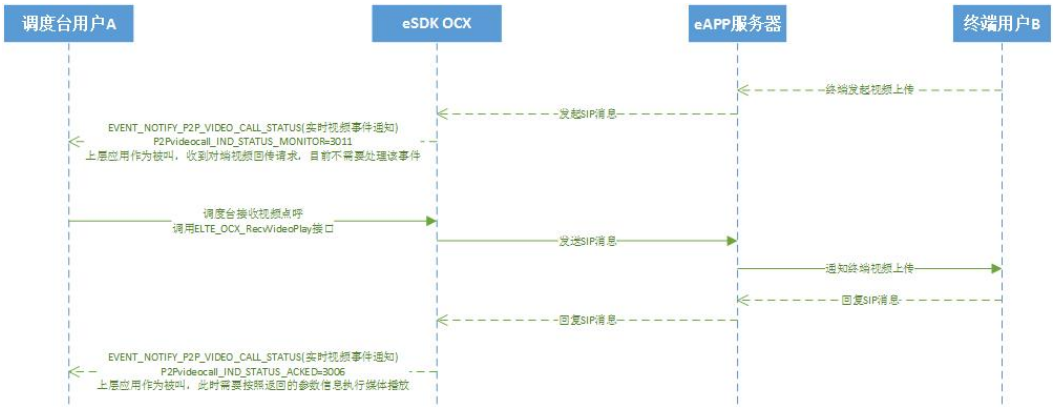
接收视频回传或视频分发

终端发起视频上传请求，SDK收到3011= P2Pvideocall_IND_STATUS_MONITOR事件通知。

说明

1. 如果第三方调度台要接收实时视频，则调用【ELTE_OCX_RecvVideoPlay(调度台接受视频回传或视频分发)】接口接收视频回传或视频分发。
2. 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_OCX_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。



终端8895向4116调度台主动发起视频回传，调度台4116收到的事件消息xml格式如下。

```
//xml code
<Content>
    <CallStatus>3011</CallStatus>
    <Callee>4116</Callee>
    <Caller>8895</Caller>
    <LocalAudioPort>0</LocalAudioPort>
    <LocalVideoPort>0</LocalVideoPort>
    <RemoteAudioPort>0</RemoteAudioPort>
    <RemoteVideoPort>0</RemoteVideoPort>
    <RemoteIp>0.0.0.0</RemoteIp>
    <Uri>8895</Uri>
    <Channel>65535</Channel>
    <SoundMute>0</SoundMute>
    <UserConfirm>9</UserConfirm>
    <Camera>0</Camera>
    <SoundPtype>-1</SoundPtype>
    <VideoFormatType>2</VideoFormatType>
    <CallID>0</CallID>
    <SignalError>-1</SignalError>
    <FromString>8895</FromString>
    <ToString>4116</ToString>
</Content>
```

接收视频回传或视频分发接口调用代码示例如下。

```
//cpp code
//接收手持终端8895的视频上传，m_strMuteType为静音类型,0为需要伴音，1为不需要伴音
CString strResult = m_eLTE_Player.ELTE_OCX_RecvVideoPlay("8895","0");
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

解析消息回调事件函数【**EVENT_NOTIFY_P2P_VIDEO_CALL_STATUS(实时视频事件通知)**】，返回的消息xml格式如下。

```
//xml code
<Content>
    <CallStatus>3006</CallStatus>
    <Callee>4116</Callee>
    <Caller>8895</Caller>
    <LocalAudioPort>64048</LocalAudioPort>
    <LocalVideoPort>64050</LocalVideoPort>
    <RemoteAudioPort>28510</RemoteAudioPort>
    <RemoteVideoPort>28514</RemoteVideoPort>
    <RemoteIp>120.9.22.172</RemoteIp>
    <Uri>8895</Uri>
    <Channel>-1</Channel>
    <SoundMute>0</SoundMute>
    <UserConfirm>9</UserConfirm>
```

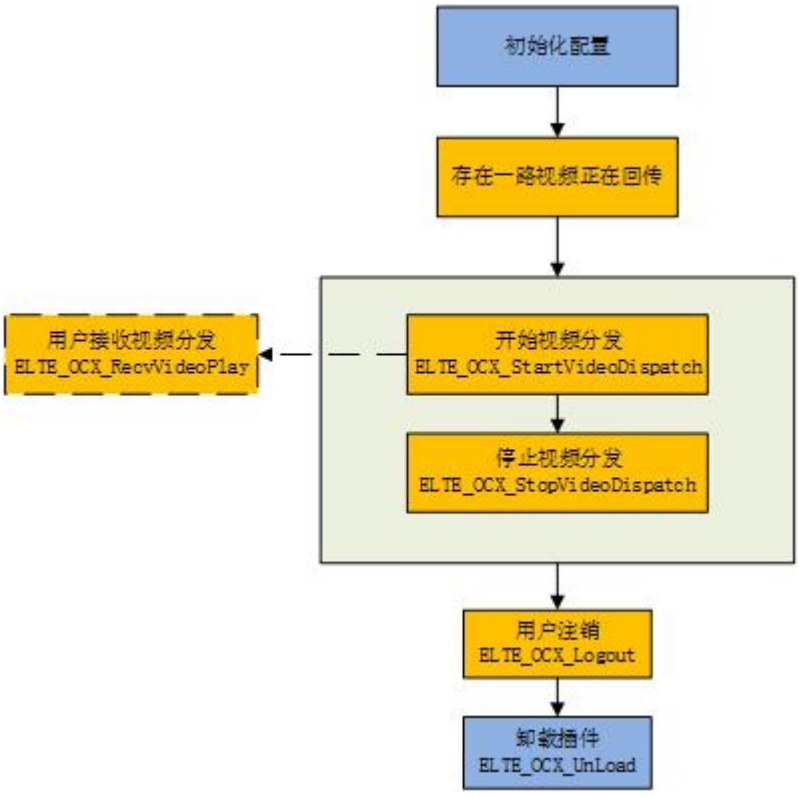
```
    <Camera>0</Camera>
    <SoundPtype>1</SoundPtype>
    <VideoFormatType>2</VideoFormatType>
    <CallID>0</CallID>
    <SignalError>-1</SignalError>
    <FromString>8895</FromString>
    <ToString>4116</ToString>
</Content>
```

当CallStatus=3006表示接收视频回传或视频分发成功，其他CallStatus值请参考【**EVENT_NOTIFY_P2P_VIDEO_CALL_STATUS(实时视频事件通知)**】。

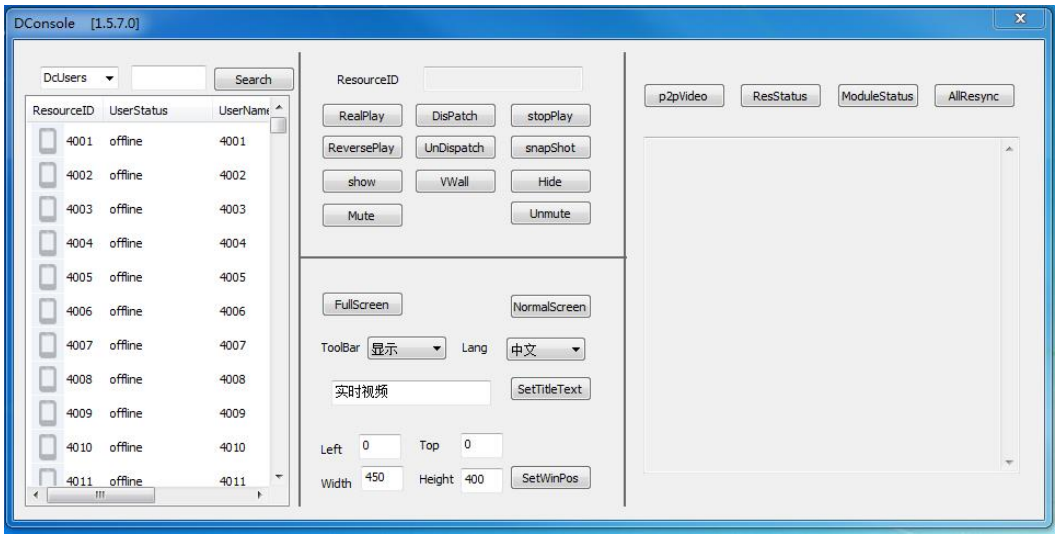
6.3.3 视频分发场景

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**，初始化配置中必须设置使用媒体面。
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_OCX_StartVideoDispatch(调度台发起视频分发)**】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.视频分发格式，取值只能为“NO”或“CIF”，其中，NO为原码转发，CIF为转码分发。

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，第一个入参是视频源ID
CString strResult =m_eLTE_Player.ELTE_OCX_StartVideoDispatch("8894",strDispatchParam);
//判断发起视频分发接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功后解析消息回调事件函数【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】，返回的消息格式如下。

```
//xml code
<Content>
    <ResourceID>4135</ResourceID>
    <ResourceName></ResourceName>
    <StatusType>22</StatusType>
    <StatusValue>4022</StatusValue>
    <AttachingGroup>0</AttachingGroup>
</Content>
```

ResourceID=4135，StatusType=22，StatusValue=4022表示调度台分发给4135后，4135已经接受视频分发。其他状态值请参考【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**。

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，需要终止其中某一终端用户的视频分发，则调用【**ELTE_OCX_StopVideoDispatch(调度台停止视频分发)**】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
CString strResult =m_eLTE_Player.ELTE_OCX_StopVideoDispatch("8894",strDispatchParam);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功后解析消息回调事件函数【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**】，返回的消息格式如下。

```
//xml code
<Content>
```

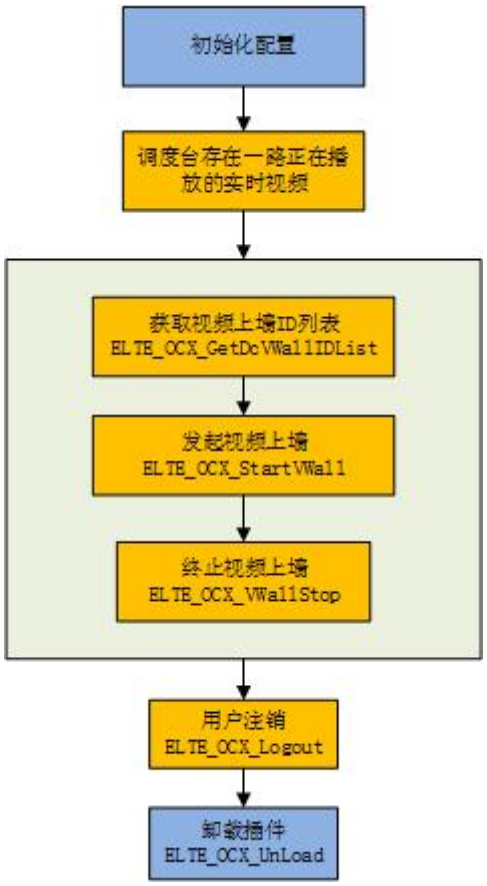
```
    <ResourceID>4135</ResourceID>
    <ResourceName></ResourceName>
    <StatusType>22</StatusType>
    <StatusValue>4023</StatusValue>
    <AttachingGroup>0</AttachingGroup>
</Content>
```

ResourceID=4135，StatusType=22，StatusValue=4023表示调度台停止视频分发给4135成功。其他状态值请参考【**EVENT_NOTIFY_RESOURCE_STATUS(资源状态变化事件通知)**。

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：



前提条件

1. 已完成**初始化配置**，初始化配置中必须设置使用媒体面。
2. 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。
3. 调度台已存在一路正在播放的实时视频。

获取视频上墙 ID 列表

视频上墙前需要先调用【**ELTE_OCX_GetDcVWallIDList(获取解码器ID列表)**】获取视频上墙的ID列表以及视频通道的状态。

接口调用代码示例如下。

```
//cpp code
//调用获取视频墙ID列表接口，入参为空
CString strResult = m_eLTE_Player.ELTE_OCX_GetDcVWallIDList();
```

接口调用结果strResult返回xml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>
    <ResultCode>0</ResultCode>
    <VWallIDList>
        <VWallID>
            <DstObjId>99910001</DstObjId>
            <IDState>1</IDState>
        <Alias>IVS1</Alias>
        </VWallID>
        <VWallID>
            <DstObjId>99910002</DstObjId>
            <IDState>1</IDState>
            <Alias>IVS2</Alias>
        </VWallID>
        <VWallID>
            <DstObjId>99910003</DstObjId>
            <IDState>1</IDState>
            <Alias>IVS3</Alias>
        </VWallID>
        <VWallID>
            <DstObjId>99910004</DstObjId>
            <IDState>1</IDState>
        <Alias>IVS4</Alias>
        </VWallID>
    </VWallIDList>
</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_OCX_VWallStart(调度台发起视频上墙)**】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。

注意

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，对于已经占用的视频墙通道必须禁止其他的视频上墙操作。

接口调用代码示例如下。

```
//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("<DstObjId>");
strVideoChannelStart.Append("99910001");
strVideoChannelStart.Append("</DstObjId>");
//保留参数
strVideoChannelStart.Append("<StrFmt>");
strVideoChannelStart.Append("</StrFmt>");
strVideoChannelStart.Append("</VideoParam>");
strVideoChannelStart.Append("</Content>");
//调用视频上墙接口，第一个参数是视频源ID
```

```
CString strResult =m_eLTE_Player.ELTE_OCX_VWallStart("8894",strVideoChannelStart);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //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。

终止视频上墙

调用【**ELTE_OCX_VWallStop(调度台终止视频上墙)**】终止视频视频上墙，入参包括视频墙通道号和待终止上墙的视频源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
CString strResult =m_eLTE_Player.ELTE_OCX_VWallStop("8894",strVideoChannelStop);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //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

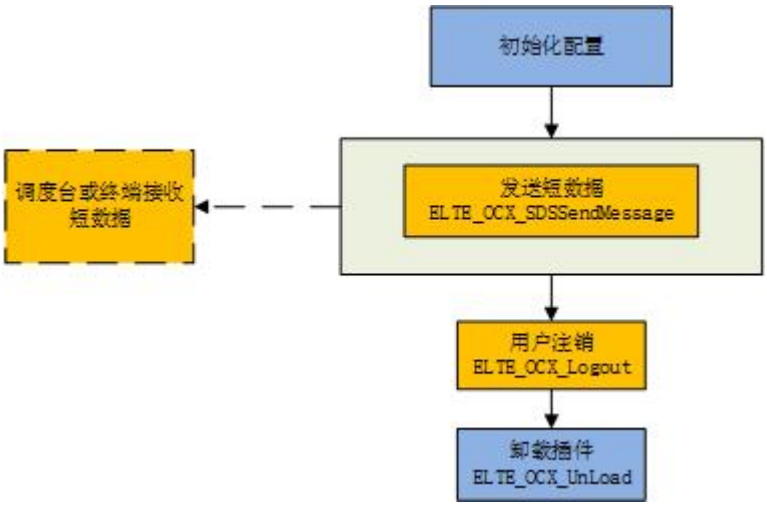
华为 eSDK eLTE SDK开放eLTE宽带集群系统的短数据业务接口，提供短数据的发送和接收接口，短数据发送【**ELTE_OCX_SDSSendMessage(发送短数据)**】使用接口，短数据的接收通过解析消息回调事件函数中【**EVENT_NOTIFY_SDS_REPORT(短数据接收上报事件通知)**】来接收短数据。

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。

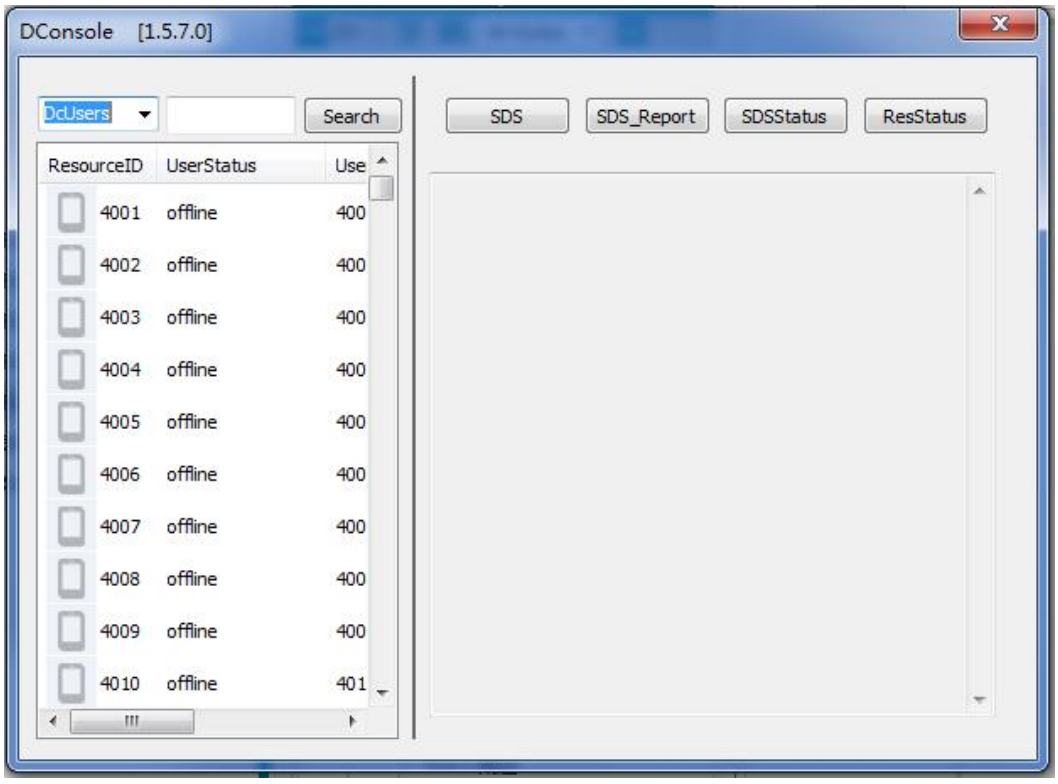
6.4.2 发送短数据场景

本章节介绍短数据发送和接收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：



短信息Sample界面：



前提条件

已完成初始化配置。

发送短数据

调用【ELTE_OCX_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。

- 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
CString strResult = m_eLTE_Player.ELTE_OCX_SDSSendMessage("4116",strMsg);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //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。

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
CString strResult = m_eLTE_Player.ELTE_OCX_SDSSendMessage("4116",strMsg);
//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

接口调用成功后，对端是否接收到了短信呢？我们需要解析消息回调事件函数中【**EVENT_NOTIFY_SDS_SEND_STATUS(短数据发送状态事件通知)**】，得到一个xml消息格式如下。

```
//xml code
<Content>
    <SdsReceiver>8895</SdsReceiver>
    <SdsSubject>1475115730892</SdsSubject>
    <SdsDirection>true</SdsDirection>
    <SdsRetCode>0</SdsRetCode>
</Content>
```

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)**】。

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。

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？

首先解析消息回调事件函数【**EVENT_NOTIFY_SDS_REPORT(短数据接收上报事件通知)**】，得到一个xml消息格式如下。

```
//xml code
<Content>
    <SdsType>0001</SdsType>
    <SdsContent>测试短信接收</SdsContent>
    <SdsFrom>8890</SdsFrom>
    <SdsSubject>1475115954492</SdsSubject>
    <SdsDirection>false</SdsDirection>
    <SdsDate>2016-09-29</SdsDate>
    <SdsTime>10:27:20</SdsTime>
</Content>
```

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

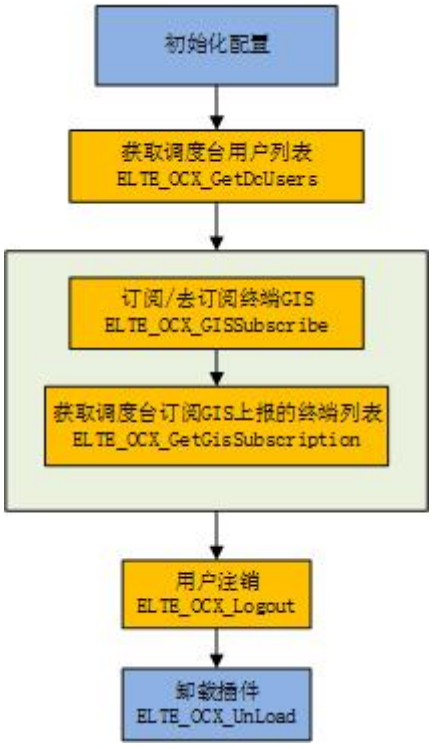
6.5.1 概述

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。调度台通过解析消息回调事件函数【**EVENT_NOTIFY_GIS_STATUS(终端GIS状态事件通知)**】来查看调度台订阅终端的状态，通过解析【**EVENT_NOTIFY_GIS_REPORT(终端GIS信息事件通知)**】来获取终端的位置信息。

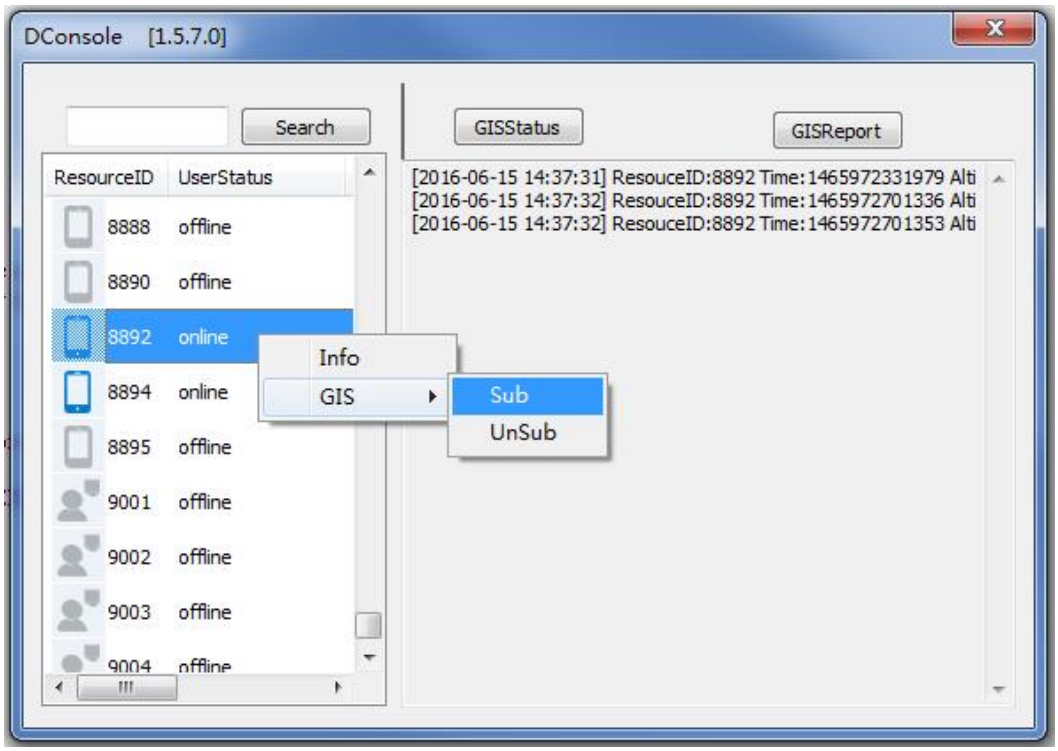
6.5.2 订阅终端 GIS 位置信息

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 Sample界面：



前提条件

1. 已完成**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。

获取调度台成员列表

参考【**获取调度台成员列表**】。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_OCX_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。

接口调用代码示例如下。

```
//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("7");
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("8890;8894;8895");
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
CString strResult = m_eLTE_Player.ELTE_OCX_GISSubscribe("0",strGISParam);
//判断接口调用是否成功//判断接口调用是否成功
if(strResult == "<Content><ResultCode>0</ResultCode></Content>")
{
    //接口调用成功
}
```

调度机eMDC是否接受调用请求需要检查消息回调事件函数中的**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**【EVENT_NOTIFY_GIS_STATUS(终端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。

```
//xml code
//<Content>
    <ResourceID></ResourceID>
    <AckStatusList>
        <AckStatus>8890:0</AckStatus>
        <AckStatus>8894:0</AckStatus>
        <AckStatus>8895:0</AckStatus>
    </AckStatusList>
</Content>
```

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【EVENT_NOTIFY_GIS_REPORT(终端GIS信息事件通知)】**得到xml消息格式如下。

```
//xml code
<Content>
    <ResourceID>8894</ResourceID>
    <Time>1475042529999</Time>
    <Altitude>15.336768773356004</Altitude>
    <Latitude>31.263462454672627</Latitude>
    <Longtitude>120.72483193140629</Longtitude>
    <TriggerCode>0</TriggerCode>
    <ReportStatus>0</ReportStatus>
</Content>
```

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_OCX_GetGisSubscription(获取调度台订阅GIS上报的终端列表)**】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
//调用获取本调度台已订阅GIS的终端列表
CString strResult = m_eLTE_Player.ELTE_OCX_GetGisSubscription("4116");
```

strResult返回值xml消息格式如下。

```
//xml code
<Content>
    <ResultCode>0</ResultCode>
    <GisQuerySubList>
        <GisQuerySubscription>
            <UeID>8890</UeID>
            <UserName>8890</UserName>
        </GisQuerySubscription>
        <GisQuerySubscription>
            <UeID>8894</UeID>
            <UserName>8894</UserName>
        </GisQuerySubscription>
        <GisQuerySubscription>
            <UeID>8895</UeID>
            <UserName>8895</UserName>
        </GisQuerySubscription>
    </GisQuerySubList>
</Content>
```

结果返回码是0，同时返回已订阅的终端列表。

7 定位指南

错误码获取方法

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。如果返回值为0，表示接口调用成功；否则表示接口调用失败，该返回值即为错误码。部分接口错误码通过解析回调的xml字串获得。

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-SDK-Windows.run.log获取。

1. B/S模式:
   接口日志文件默认生成在C:\eSDK_eLTE\eLTE_NativeService\log\目录下
2. C/S模式:

接口日志文件默认生成在加载资源包所在路径的同一级目录下的log/目录下

- **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、接口入参以及调用结果。

以登录eLTE_OCX_Login接口为例

目录下的eSDK-eLTE-OCX.run.log内容：

2016-06-13 18:51:56 743|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Login} --- ENTRY

2016-06-13 18:51:56 754|ERROR|[3668]|DcLogin failed.

2016-06-13 18:51:56 754|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Login} --- LEAVE

对应eSDK-eLTE-SDK-Windows.run.log下内容为：

2016-06-13 18:51:56 744| INFO|[6892]|userId = 4119, serverIP = 172.22.9.105, localIP = 172.24.4.239, sipPort = 5060

2016-06-13 18:51:56 745|ERROR|[6892]|invokeOp DC_LOGIN failed. (-40134)

IVS_OCX_Login：对应时间信息即可以查看该接口调状态信息

-40134：错误码

错误信息查询方法

接口参考**eSDK eLTE OCX 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

- B/S模式:

  先加载OCX控件，然后将C:\eSDK_IVS\config\log.xml中的DebugLogSwitch开关设置为True，表示开启Debug日志。默认为False，表示关闭Debug日志。然后重启IE，调用IVS_OCX_Init()初始化接口。

- C/S模式:

  加载资源包路径下的config\log.xml中的DebugLogSwitch开关设置为True，表示开启Debug日志。默认为False，表示关闭Debug日志。重启程序。

获取Debug日志

- B/S模式:

  Debug日志文件默认生成在C:\eSDK_eLTE\eLTE_NativeService\log\目录下

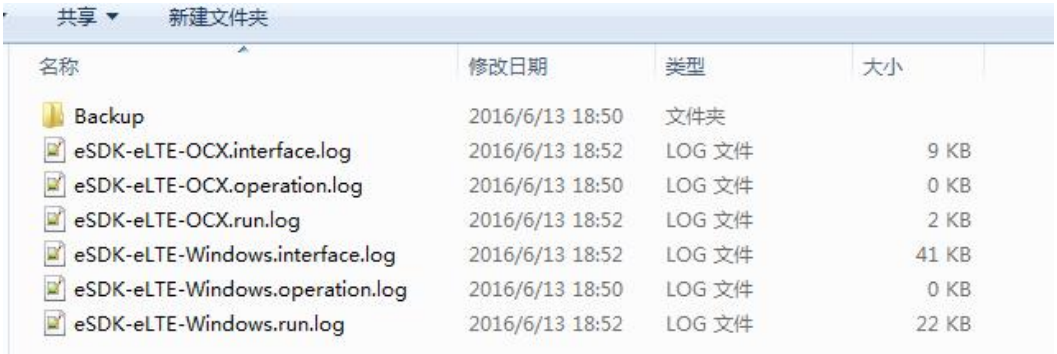
- C/S模式:

  加载资源包所在路径下的log\

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_OCX_SetLogPath接口自定义日志路径

log文件夹下共包括OCX，API两组日志，每组日志分成接口调用，操作日志，运行日志三类，如下截图：



以.log后缀前的结尾描述进行区分：

interface：调用接口(OCX/API)日志信息

operation：操作日志信息

run：运行日志信息

开启debug日志后，sdk\debug目录下的xxxx.run.log为接口调用产生的sdk debug日志，此日志信息最丰富，是定位问题最关键的日志

日志解析

1. 全局搜索关键字“ELTE_OCX_”，搜索结果即为全部调用的OCX接口，如下截图

```
16-06-13 18:51:50 617| CRIT|[3668]==============log start============
16-06-13 18:51:50 617| INFO|[3668]Upload timer starts. | IP:esdk-log.huawei.com
16-06-13 18:51:50 617|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Load} --- ENTRY
16-06-13 18:51:50 617| INFO|[3668]|ulType = 1
16-06-13 18:51:56 694| INFO|[3668]|snapshotFormat = 1, savePath =
\eLTE_C60\Client_Ext\test\eLTE_Player\eLTE_PlayerDemo\eLTE_PlayerDemo\..\output\debug\.\snapshot\
16-06-13 18:51:56 694| INFO|[3668]|dwVolume = 100
16-06-13 18:51:56 694| INFO|[3668]|dwVolume = 100
16-06-13 18:51:56 705| INFO|[3668]|Gdiplus::Image::FromFile failed.
16-06-13 18:51:56 742|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Load} --- LEAVE
16-06-13 18:51:56 743|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Login} --- ENTRY
16-06-13 18:51:56 754|ERROR|[3668]|DcLogin failed.
16-06-13 18:51:56 754|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Login} --- LEAVE
16-06-13 18:51:56 758| INFO|[3668]|iEventId = 8
16-06-13 18:51:56 759| INFO|[3668]|iEventId = 8
16-06-13 18:51:56 759| INFO|[3668]|iEventId = 8
16-06-13 18:51:56 759| INFO|[3668]|iEventId = 8
16-06-13 18:51:56 759| INFO|[3668]|iEventId = 8
16-06-13 18:51:58 804|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Logout} --- ENTRY
16-06-13 18:51:58 815|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Logout} --- LEAVE
16-06-13 18:51:58 835|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_UnLoad} --- ENTRY
16-06-13 18:51:59 446| CRIT|[3668]==============log end==============
```

2. 在进入和退出每一个接口时，都会打印一条Enter和Leave日志。Enter日志一般都会打印一些入参信息，Leave日志打印表示退出该接口。

   2016-06-13 18:51:56 705| INFO|[3668]|Gdiplus::Image::FromFile failed.

   2016-06-13 18:51:56 742|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Load} --- LEAVE

   2016-06-13 18:51:56 743|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Login} --- ENTRY

   2016-06-13 18:51:56 754|ERROR|[3668]|DcLogin failed.

   2016-06-13 18:51:56 754|DEBUG|[3668]|{CeLTE_PlayerCtrl::ELTE_OCX_Login} --- LEAVE

3. 日志分析：（以红色部分为例分析）：

2016-06-13 18:51:56 754|ERROR|[3668]|DcLogin failed.

2016-06-13 18:51:56 754：日志打印时间

error：日志级别

DcLogin failed：产生错误描述

Enter/Leave：进入/退出IVS_OCX_接口

对应到eSDK-eLTE-SDK-Windows.run.log日志文件内：

2016-06-13 18:51:56 745|ERROR|[6892]|invokeOp DC_LOGIN failed. (-40134)

2016-06-13 18:51:56 745：日志打印时间

error：日志级别

invokeOp DC_LOGIN failed.：产生错误描述

-40134：错误码

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
2015-10-25	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。