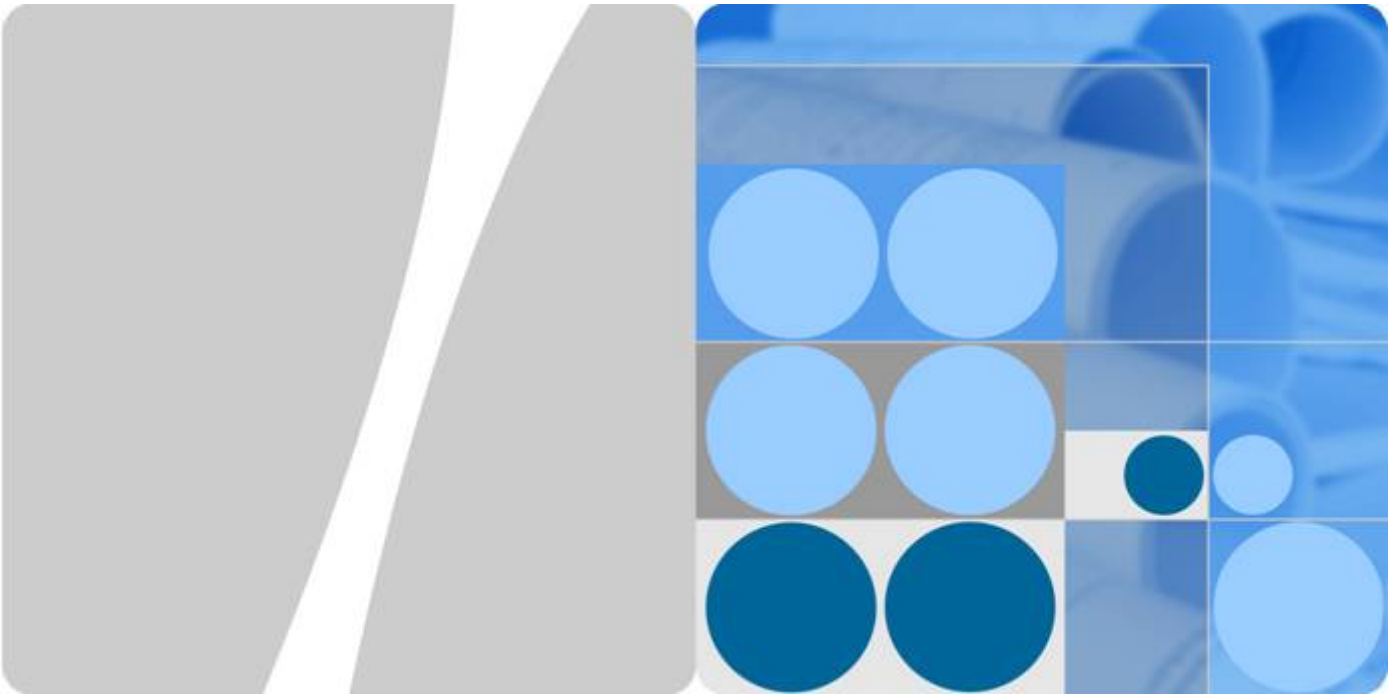


eSDK TP V100R005C70 开发指南 01 (服务端 Native,C++)

文档版本　01

发布日期　2016-08-28

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 TP for C++是什么

华为eSDK TP for C++是针对华为视频会议（Video Conference，VC）解决方案的业务管理平台（SMC2.0），基于标准C++ API，为合作伙伴提供统一账号管理、视频会议调度、会议控制等业务能力。

丰富的开放能力使得合作伙伴易于将华为视频会议解决方案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、便捷以及实用的视频会议调度平台。

eSDK TP for C++提供什么

C++ API

视频会议二次开发使用的C++ API，提供dll库文件和接口参考文档。更多信息请参见**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，完成视频会议业务开发。更多信息请参见**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 TP二次开发提供基于各种语言的源代码。更多代码资源请前往**Github**获取。

关于 SMC2.0

SMC2.0提供华为视讯设备管理和业务控制功能，可根据企业组织架构进行分级分权管理。

系统管理员可对整网设备进行统一管理和维护；普通用户可通过SMC2.0提供的丰富会议召集和控制功能，尽情享受视频会议带来的沟通之乐。

更多信息请参见**SMC2.0**。

2 内容导航

开发指南有什么

本开发指南告诉您如何安装和配置环境、使用C++ API调用SMC2.0平台的业务功能，以及华为eSDK提供的开发者支持服务。主要内容如下：

1. **相关资源**：二次开发过程中可能涉及到的软件、文档资源链接、技术支持。包括如何通过社区网站获取资料、Sample Codes下载链接、介绍C IDE插件及使用方法、如何使用远程实验室等。
2. **Hello World**：如果您仅仅是尝试把SDK运行起来，应该首先看这部分内容，它会详细说明如何下载及安装SDK，以及如何配置您的开发环境，并协助您迅速调通第一个接口。
3. **初始化配置**：在开发业务功能之前需要完成的初始化配置，这些配置动作仅需执行一次。
4. **典型场景开发**：介绍eSDK TP典型功能场景，包括开发流程、样例代码以及注意事项等。
5. **问题定位**：开发过程中常见问题的定位方法。
6. **修订记录**：各版本开发指南更新细节。

阅读建议

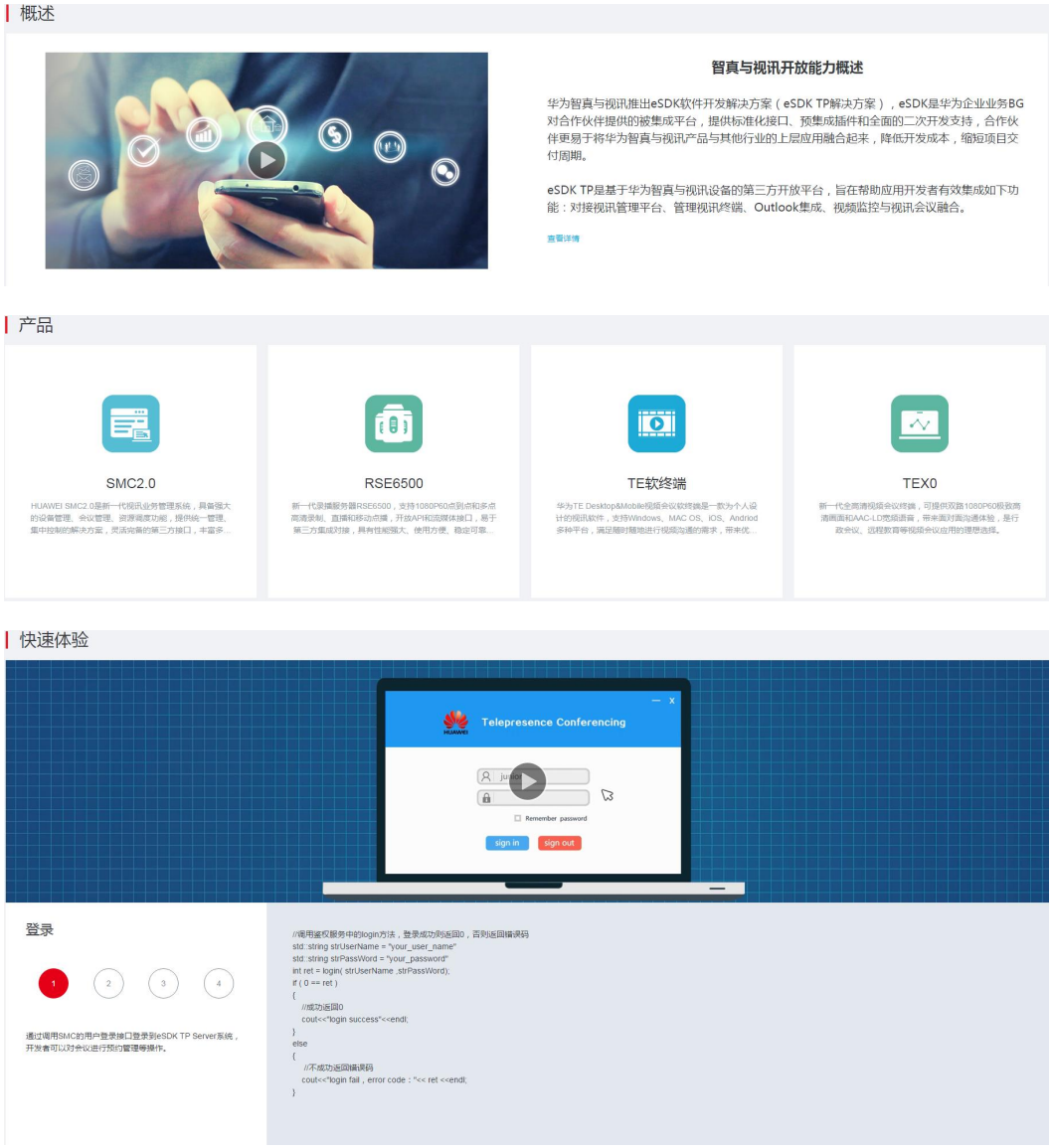
- 如果只是想快速入门，可仅参考**Hello World**章节。
- 如果想深入eSDK TP核心业务的二次开发，建议您参**考典型场景开发**章节。
- 在使用SDK过程中遇到问题可以参考**问题定位**章节进行自助定位，或者前往华为开发者社区TP版块**在线FAQ查询**，或者求助**eSDK TP技术支持**。

本开发指南配套的SDK版本号为**V1.5.70**，如果需要查看其他版本SDK对应的开发指南，请进入**华为开发者社区资源中心**。

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**智真与视讯**页面，快速体验视频会议业务功能，获取eSDK TP的二次开发SDK工具包以及技术支持等。



3.2 SDK 下载路径

进入**华为开发者社区资源中心**，点击“**SDK > 企业云通信 > SMC2.0 > eSDK TP C++ SDK**”，选择对应版本进行下载。

推荐使用最新的版本V1.5.70。

3.3 Sample Codes

这里建议您使用Visual Studio 2010及以上版本编译执行Sample Codes。

Sample Codes列表包含了典型场景开发章节中介绍的开发场景Demo源码路径，演示如何调用eSDK TP接口。

Sample Codes列表如下：

名称	说明
eSDK_TP_LOGIN_DEMO	包括登录eSDK TP Server、保活以及登出等功能
eSDK_TP_HOLDCONF_DEMO	包括获取会场列表、预约普通会议、预约周期会议、查询已调度的会议、查询会议中的会场状态、查询会场信息等功能
eSDK_TP_ADHOCCONF_DEMO	包括创建Ad hoc会议模板、查询Ad hoc会议模板、创建Ad hoc会议等功能
eSDK_TP_CONFCTRL_DEMO	包括指定会场视频源、设置会议多画面参数、设置广播多画面等功能

如果需要下载SDK及尝试调用接口，请参考Hello World。

3.4 接口参考

接口参考主要包含以下内容：

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。

数据类型：详细说明eSDK对外提供的自定义的数据类型（包括类、结构体、枚举、等），例如：

- 数据类型名。
- 对有继承/嵌套关系的数据结构进行说明，如结构体嵌套关系，要提供总的嵌套关系表（要包括基本数据类型）。
- 包括的数据成员以及数据成员的含义。

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。

3.5 IDE 插件

我们提供配套Eclipse和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TP Demo工程，以及自动安装TP SDK包，二次开发效率更高。IDE插件的详细使用说明，请参考**IDE快速入门**。

3.6 免费使用远程实验室

华为 eSDK 远程实验室简介

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

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

目前，华为远程实验室已发布云计算、SDN、大数据、统一通信与协作、BYOD、eLTE、敏捷网络7个生态圈的26个实验室环境。

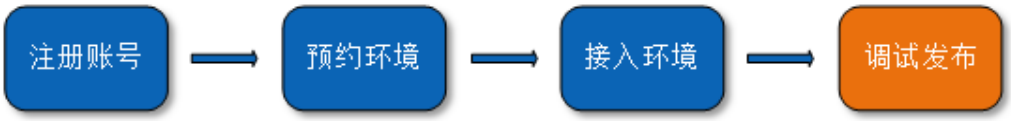
相关信息请查询**华为开发者社区远程实验室**。

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。

如何免费使用远程实验室

免费使用远程实验室的步骤如下：



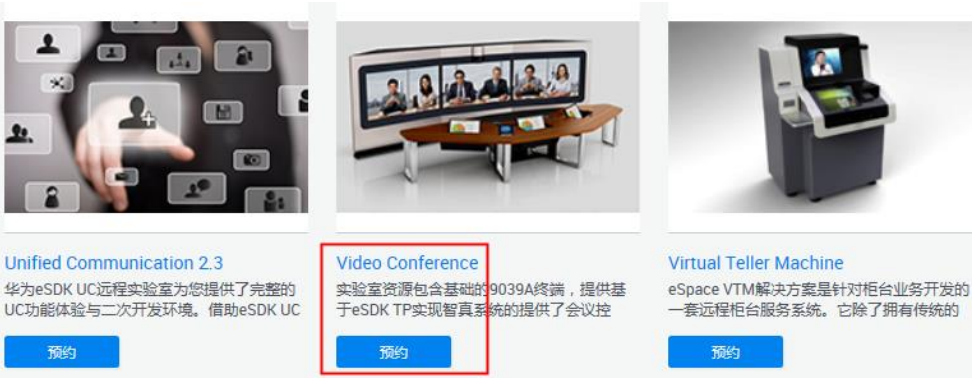
步骤1 注册账号。

1. 点击**http://developer.huawei.com/cn/ict/remotelab**，进入华为远程实验室网站。
2. 若已有华为注册账号，可直接在**华为远程实验室网站**使用已注册账号登录。
3. 若还未注册华为账号，请在**华为远程实验室网站**点击“**进入实验室**”，跳转至账号注册页面填写注册信息，并在注册邮箱中激活您的华为账号。

请使用IE10+（或最新版本的Chrome/Firefox/Safari等）浏览器接入使用，以获得最佳体验。如果您的浏览器版本低于要求的版本，部分页面可能无法打开。

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，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内容页面，状态为已登录。使用视频会议调试环境时，在右侧“分类浏览”项下选择“**UC&C**”，在筛选出的“环境目录”项下选择“**Video Conference**”，点击“**预约**”按钮，如下图所示。



默认预约环境的可用时长为2小时，用户可以自定义预约时长，最多可支持4小时。

3. 预约成功后，系统会自动发送VPN网关地址、用户名、密码等信息至您的注册邮箱中，根据邮件中的提示信息下载SSL-VPN客户端软件，完成本地安装。后续步骤需要使用该信息登录VPN客户端软件进行远程实验室环境连接。

步骤3 接入环境。

1. 若您已成功接入华为远程实验室环境，请忽略此步骤。
2. 双击桌面的“SVNClient”运行VPN客户端软件，将远程实验室预约成功后邮件通知中发送的VPN网关地址及用户名、密码等信息，分别输入到VPN客户端对应的输入框内。

3. 点击“**登录**”，系统会启动远程实验室连接，当桌面右下角出现SSL-VPN连接标识符时，系统会提示VPN连接成功。

步骤4 获取账号、密码。

点击拓扑图中的  图标，在出现的  中单击属性图标  ，在右侧弹出的属性窗口中获取账号、密码。

步骤5 调测发布。

使用申请成功的视频会议账号、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会逐渐地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了解不同版本有哪些更新。具体的更新信息包含：

- SDK名称
- 产品名称：SDK配套的产品名称。
- 更新时间：SDK更新发布上线的时间。
- 版本号：SDK当前的版本号。
- 下载链接：链接中包含SDK Demo及配套文档。
- 更新记录描述：描述变更的特性。

3.8 技术支持渠道

DevCenter 提单

在开发过程中，您有任何问题均可以至**DevCenter**中提单跟踪。具体方法如下：

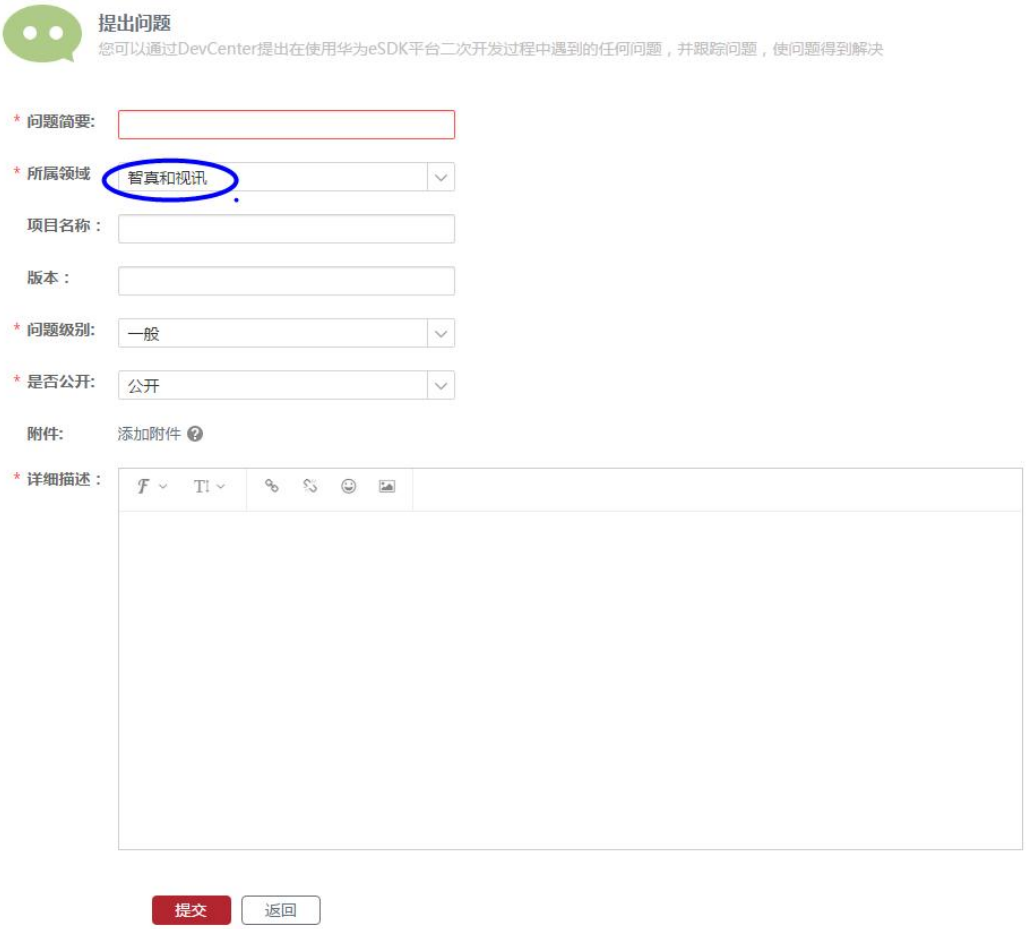
步骤1 登录**DevCenter**首页。

已注册有华为开发者社区账号的，可直接登录。未注册的按照相关要求注册后登录。

步骤2 选择“问题 > 新建问题”，进入创建问题单页面。

步骤3 填写问题单信息。

逐项填写问题单信息，包括问题简要、所属领域、项目名称、版本、问题级别、是否公开等。其中，带*号为必填选，所属领域选择“统一通信与协作 > 智真和视讯”。



步骤4 点击“提交”，完成问题单的创建。

创建完成后，可至**DevCenter**首页查看已创建的问题单，并进行跟踪管理。

----结束

其他渠道

如果您在远程实验室使用过程中有任何疑问，可以通过以下方式联系技术支持人员：

- 华为技术支持热线电话：400-822-9999（转二次开发）
- 华为技术支持邮箱：esdk@huawei.com

4 Hello World

4.1 Hello World 开发流程

本示例以C++语言进行eSDK TP的二次集成开发。

开发过程中的故障处理请参考**问题定位**。

Hello World实现流程如下：

4.2 准备环境

开发工具

- 操作系统：Windows 7专业版。
- Microsoft Visual Studio：Visual Studio 2010专业版。

SDK 软件包

SDK软件包名称：**eSDK_TP_Native_V1.5.70_CPP.zip**

SDK软件包下载路径：参见**SDK下载路径**。

SDK软件包解压缩后包含的文件如下：

- **bin**目录：包含可执行程序在debug模式下和release模式下所依赖的库及配置文件。
- **include**目录：包含SDK提供的头文件。
- **lib**目录：包含程序在debug模式下和release模式下编译所依赖的静态数据链接库。
- **pdb**目录：包含程序在debug模式下和release模式下调试代码所需的程序数据库。

eSDK TP 服务端

使用eSDK TP提供的C++ SDK，需要部署eSDK TP服务端，获取WebService服务地址和鉴权账号。

1. 如果eSDK TP服务端已经部署，请直接向管理员获取服务地址、鉴权账号和密码。

   说明

   根据eSDK安装后选择的不同鉴权方式，eSDK TP账号及密码对应说明如下：

   - 如果使用透传鉴权，此处应用账号和密码分别为在智真设备配置的登录名和密码。
   - 如果使用单用户鉴权或一对一映射鉴权，此处应用账号和密码为第三方应用向eSDK鉴权的用户名和密码。
   - 关于鉴权策略及应用账号和密码的具体说明，请参见**《eSDK TP V100R005C70 安装配置指南》**。

   eSDK TP服务端提供服务的URI为：**http(s)://*${ip}*:*${port}*/esdk/services**。其中，“${ip}”为部署eSDK TP服务端的IP地址，“${port}”为部署eSDK TP服务端的端口号。

2. 如果eSDK TP服务端没有部署，请进入**华为开发者社区资源中心**，点击“**SDK > 企业云通信 > SMC2.0 > 平台安装包**”获取平台安装包和TP业务包，参考**《eSDK TP V100R005C70 安装配置指南》**完成eSDK TP服务端的安装部署。
   - eSDK平台安装包：
     - 若是Windows系统（32位），请下载eSDK_Platform_V1.5.70_Windows_x86.zip
     - 若是Windows系统（64位），请下载eSDK_Platform_V1.5.70_Windows_x64.zip
     - 若是Linux系统（64位），请下载eSDK_Platform_V1.5.70_Linux_x64.zip
   - eSDK TP业务包：**eSDK_TP_V1.5.70.zip**

智真与视讯产品

- SMC：HUAWEI SMC2.0 V500R002C00SPC200、HUAWEI SMC2.0 V500R002C00SPC100
- MCU9650：VP9660 V500R002C00SPC200

若您还未安装、配置eSDK TP服务端，未购买华为智真与视讯产品，可免费使用远程实验室进行软件编译及调试。

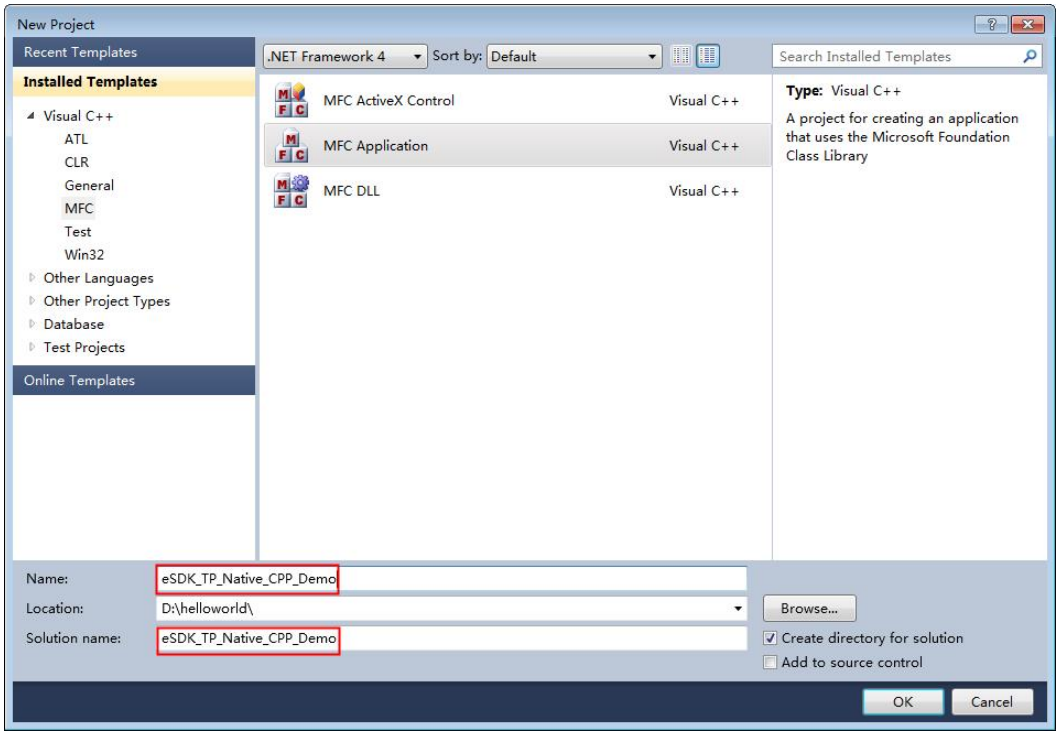
如果您在下载软件或代码样例的过程中遇到异常，请联系华为公司eSDK TP技术支持寻求帮助。

4.3 创建工程

步骤1 新建Visual Studio工程项目。

1. 打开Microsoft Visual Studio 2010，选择“**File > New > Project**”。系统显示“New Project”界面。
2. 选择“**Visual C++ > MFC > MFC Application**”，在“**Name**”对应的文本框中输入新建工程项目名称“eSDK_TP_Native_CPP_Demo”，在“**Location**”对应的文本框中选择工程存放路径。如**图4-1**所示。

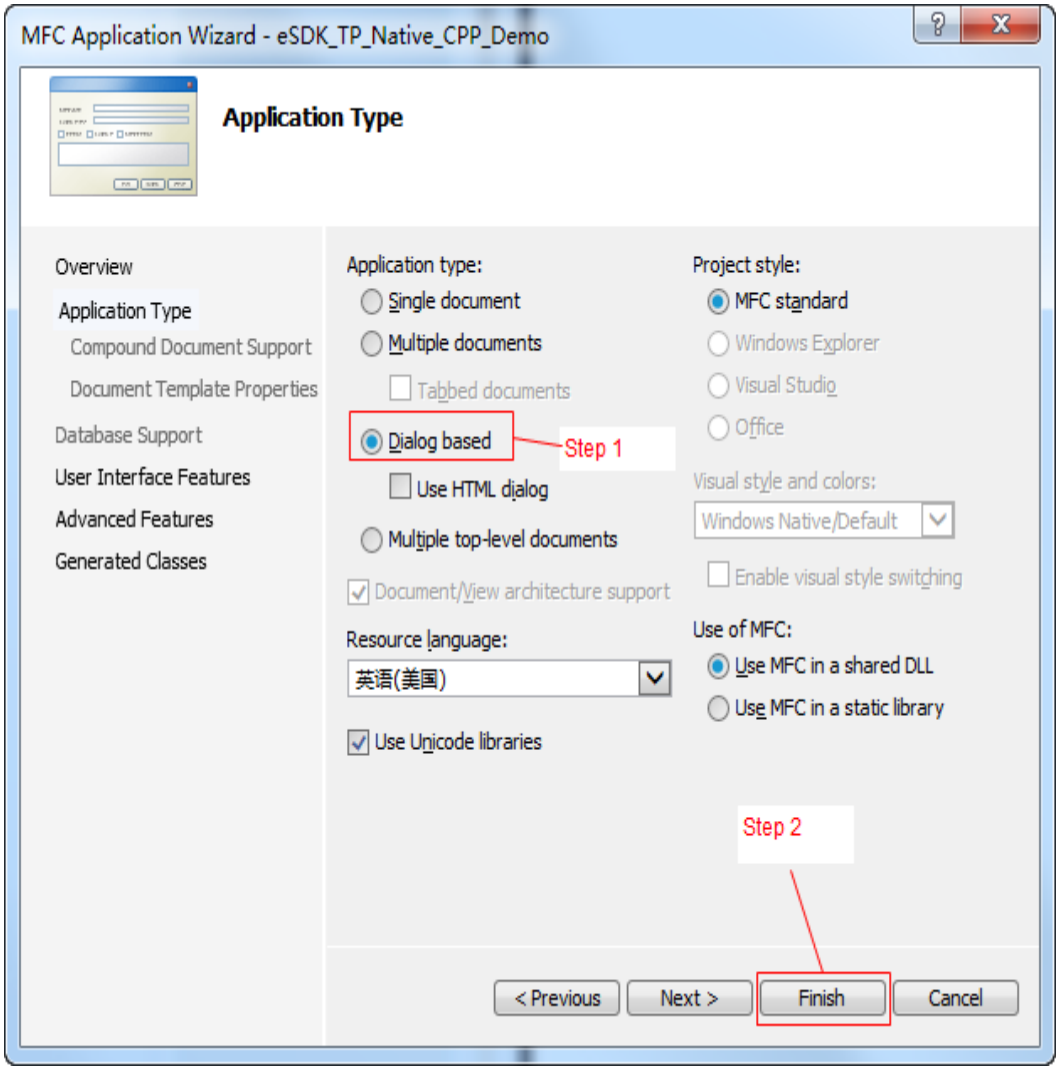
图 4-1 New Project 界面



用户可以自定义工程项目名称和存放路径。

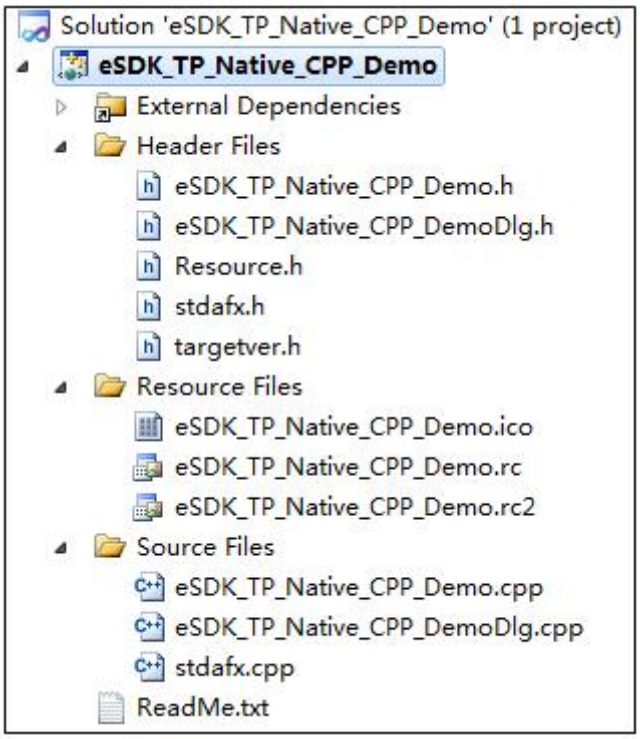
3. 单击“**OK**”，创建eSDK_TP_Native_CPP_Demo工程，在弹出的“Application Wizard”界面中，单击“**Next**”按钮。
4. 在图4-2中的“Application Type”栏选择“Dialog based”选项，其他选项保持默认，单击“**Finish**”按钮。

图 4-2 Application Type 界面



新创建的工程主界面如图4-3所示。

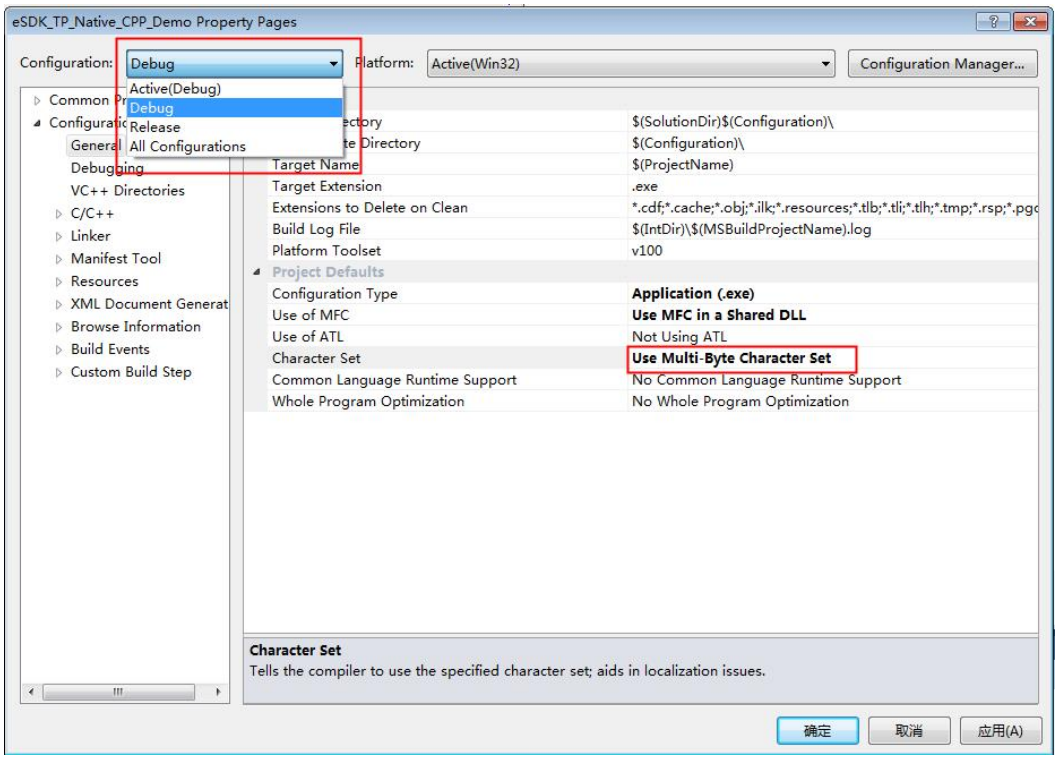
图 4-3 工程主界面



步骤2 设置eSDK_TP_Native_CPP_Demo工程的编码格式。

1. 在eSDK_TP_Native_CPP_Demo工程上单击右键，选择“**Properties**”，默认打开“Property Pages”窗口。
2. 选择所需改动的配置模式：Debug模式或Release模式，如图4-4所示。
3. 选择“**Configuration Properties > General**”，在“Character Set”选项中选择“Use Multi-Byte Character Set”项，如图4-4所示。

图 4-4 编码格式设置界面



4. 单击“确定”完成编码格式设置。

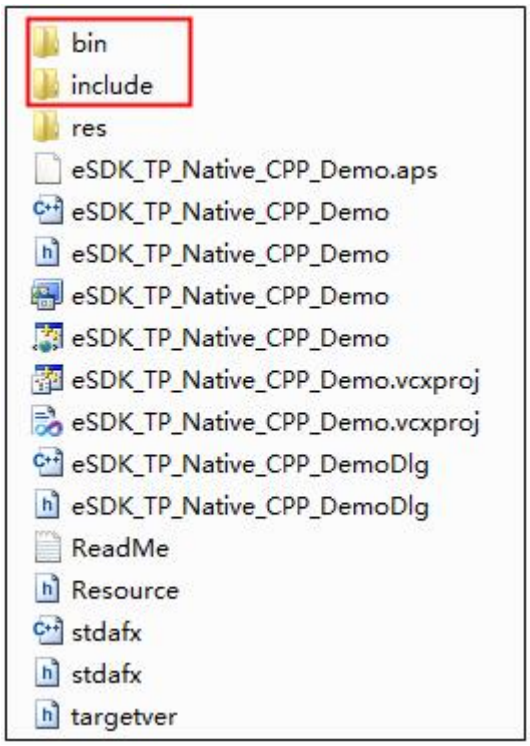
----结束

4.4 加载头文件和动态链接库

在Visual Studio 2010中导入Native的头文件和动态链接库。

步骤1 解压SDK软件包，将**include**文件夹和**lib**文件夹放到eSDK_TP_Native_CPP_Demo工程目录下。

图 4-5 eSDK_TP_Native_CPP_Demo 工程目录

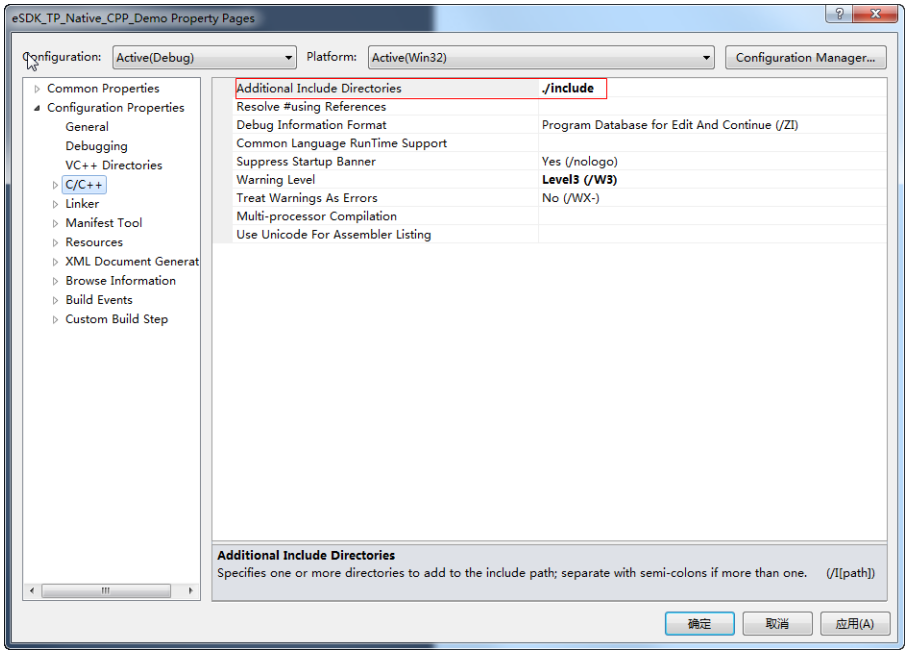


步骤2 在eSDK_TP_Native_CPP_Demo工程上单击右键，选择“**Properties**”，默认打开“Property Pages”窗口。

步骤3 选择所需改动的配置模式：Debug模式或Release模式。

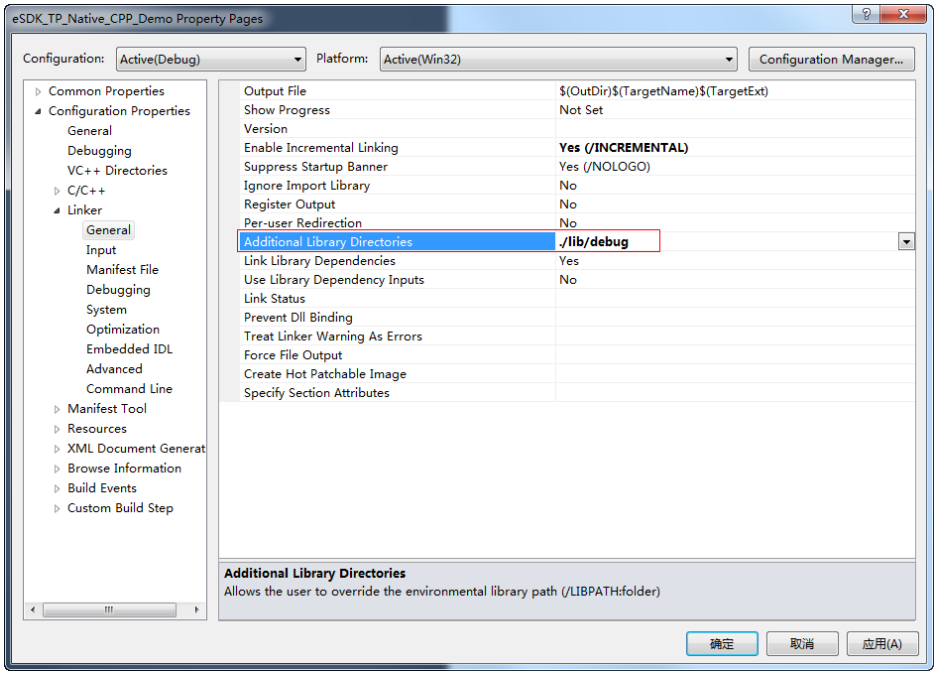
步骤4 选择“**Configuration Properties > C/C++ > General**”，在“Additional Include Directories”输入框中手动输入SDK头文件所在的路径。如果SDK头文件放在图4-5所示的路径下，填写“**./include**”。

图 4-6 设置 SDK 库依赖的头文件界面



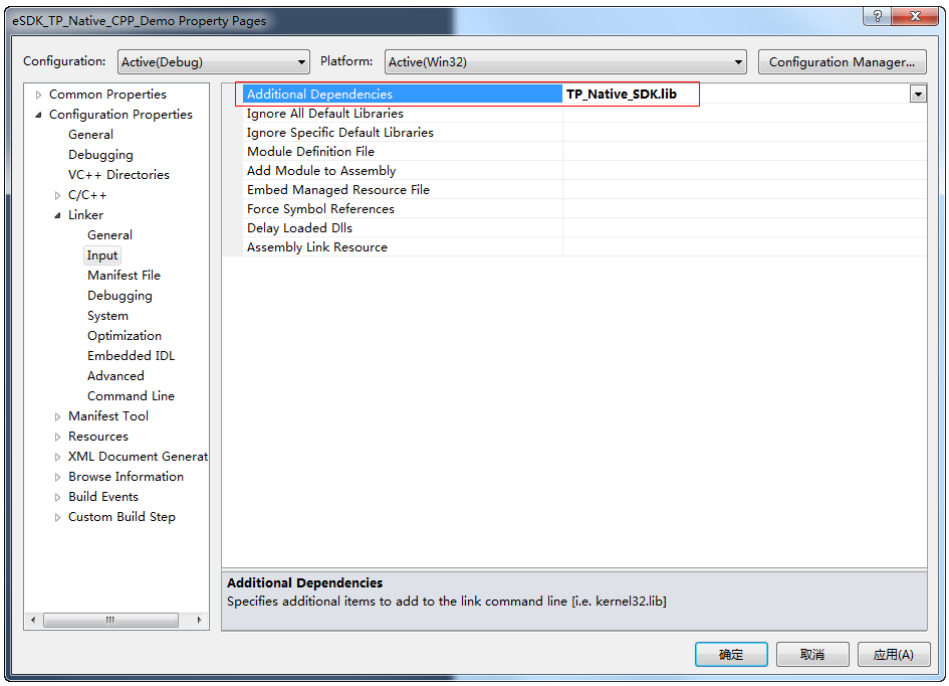
步骤5 选择“**Configuration Properties > Linker > General**”，在“Additional Library Directories”输入框中手动输入SDK的lib文件路径。如果lib文件放在图4-5所示的路径下，在Debug模式下填写“**./lib/debug**”，在Release模式下填写“**./lib/release**”。

图 4-7 设置 SDK 静态数据链接库路径界面



步骤6 选择“**Configuration Properties > Linker > Input**”，在“Additional Dependencies”输入框中手动输入eSDK_TP_Native_CPP_Demo所依赖的SDK的lib文件名：“**TP_Native_SDK.lib**”。

图 4-8 设置 SDK 静态数据链接库名称界面



步骤7 单击“**确定**”，完成头文件和动态链接库的加载。

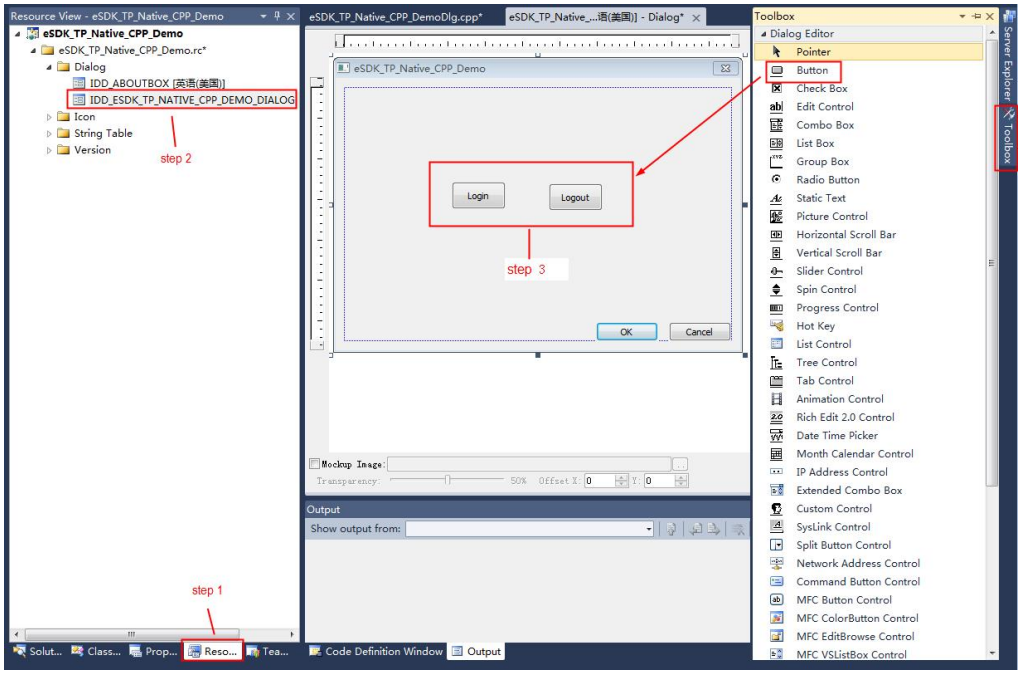
----结束

4.5 设计界面布局

设计eSDK_TP_Native_CPP_Demo窗体界面并配置url。

步骤1 双击已创建的eSDK_TP_Native_CPP_Demo工程，选择“**View > Resource View**”，在“Resource View”窗口中双击“IDD_ESDK_TP_NATIVE_CPP_DEMO_DIALOG”，打开“**eSDK_TP_Native_CPP_Demo**”窗体。

步骤2 选择“**菜单 > View > Toolbox > Button**”，添加两个按钮至“**eSDK_TP_Native_CPP_Demo**”窗体中。右键单击按钮，选择“**Properties > Caption**”，将两个按钮的Caption属性分别修改为“**Login**”和“**Logout**”。



----结束

4.6 编码实现

1. 引入头文件。

```
#include "TP_Native_SDK.h"
#include "TP_Native_Types.h"
```

2. 调用login接口。

双击“eSDK_TP_Native_CPP_Demo”窗体上的“**Login**”按钮，在代码编辑框内输入相关代码，参考如下：

```
void CeSDK_TP_Native_CPP_DemoDlg::OnBnClickedLogin()
{
    int ret = TP_E_RET_CODE_FAIL;
    /* input eSDK Server username and password,if it called succeed, it will return 0 */
    ret = login( "username", "password" );
    if ( TP_E_RET_CODE_SUCCESS == ret )
    {
        AfxMessageBox( "Login successful." );
```

```
    }else  {
        char result[1024] = { 0 };
        sprintf_s( result, 1023, "Login failure;ErrCode : %d.", ret );
        AfxMessageBox( result );
    }
}
```

3. 调用logout接口。

   双击“eSDK_TP_Native_CPP_Demo”窗体上的“**Logout**”按钮，在代码编辑框内输入相关代码，参考如下：

```
void CeSDK_TP_Native_CPP_DemoDlg::OnBnClickedButton3()
{
    int ret = TP_E_RET_CODE_FAIL;
    /* if it called succeed, it will return 0 */
    ret = logout();
    if ( TP_E_RET_CODE_SUCCESS == ret )
    {
        AfxMessageBox( "Logout successful." );
    }else  {
        char result[1024] = { 0 };
        sprintf_s( result, 1023, "Logout failure;ErrCode : %d.", ret );
        AfxMessageBox( result );
    }
}
```

4.7 编译及调试

有华为 SMC 环境

若您已购买华为视讯产品，请直接在eSDK TP Server上配置产品的用户名、密码以及IP地址，按照下面介绍的方法**调试运行**程序。

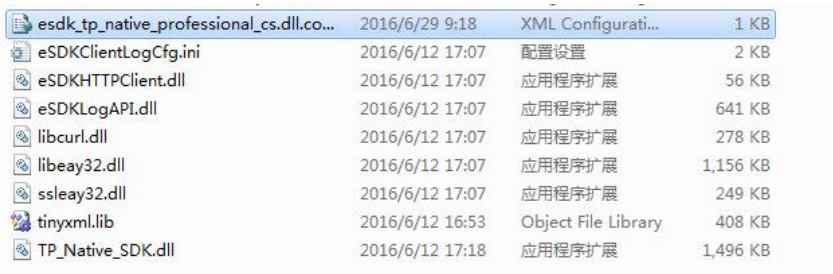
无华为 SMC 环境

若您还未购买华为视讯产品，可登录**华为远程实验室**，**免费**申请SMC环境，按照下面介绍的方法**调试运行**程序。

调试运行

步骤1 打开 eSDK_TP_Native_CPP_Demo 工程“Debug”目录或“Release”目录下的配置文件“esdk_tp_native_professional_cs.dll.config”，修改url中的value值，具体操作请参见初始化配置。

步骤2 将eSDK包中“bin\debug”目录下的动态库和配置文件放到eSDK_TP_Native_CPP_Demo工程生成的执行文件目录下。



步骤3 这里以调试login、logout接口为例。

1. 编译程序：菜单>Build>Build Solution(图1所示)。当Output对话框出现Build： 1 succeeded(图2所示)，表示编译成功。

图 4-9

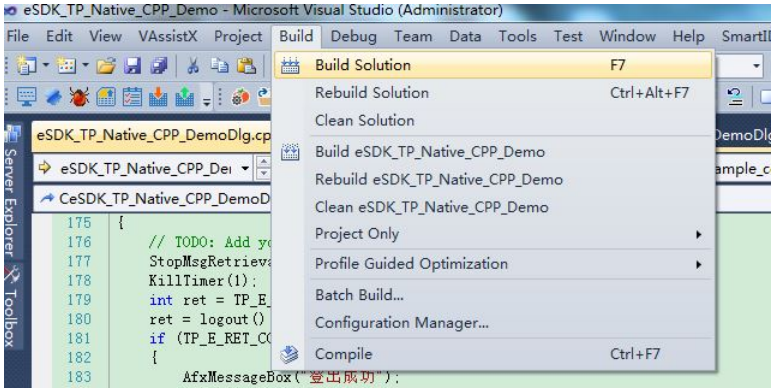
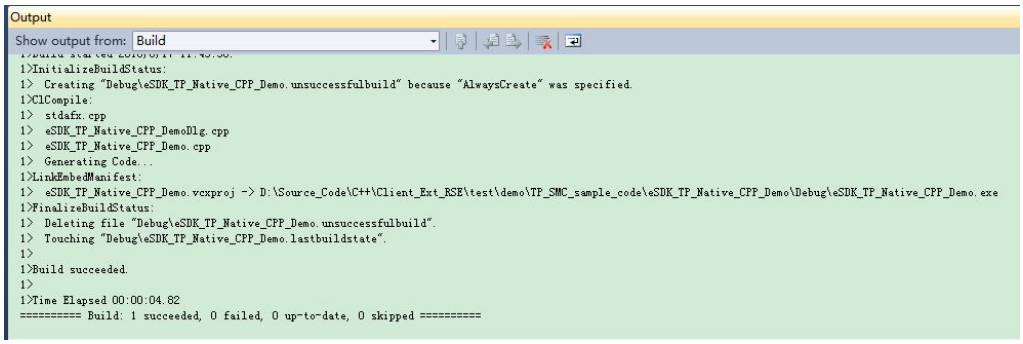


图 4-10



2. 运行程序：菜单>Debug>Start Debugging(图3所示)。当出现程序的对话框，表示运行程序成功(图4所示)

图 4-11

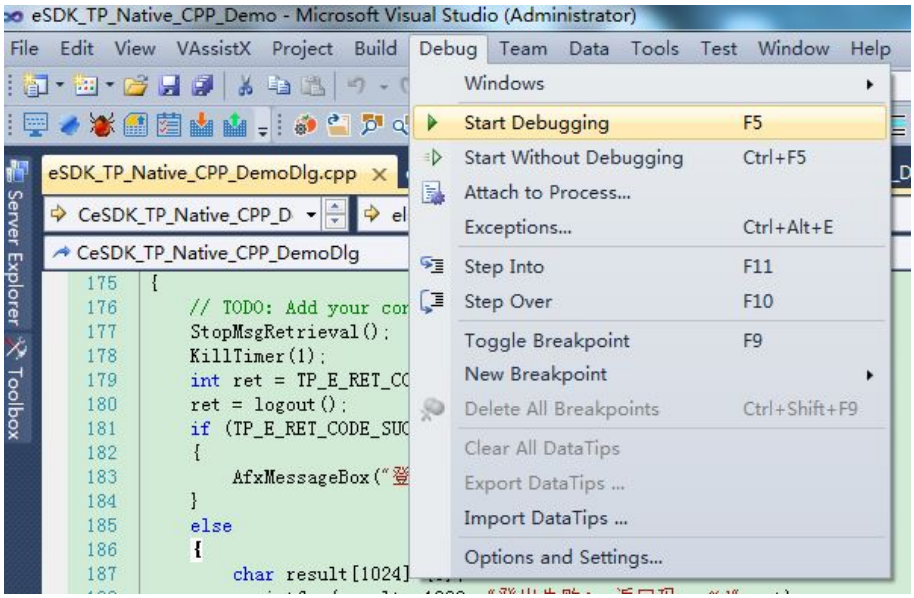
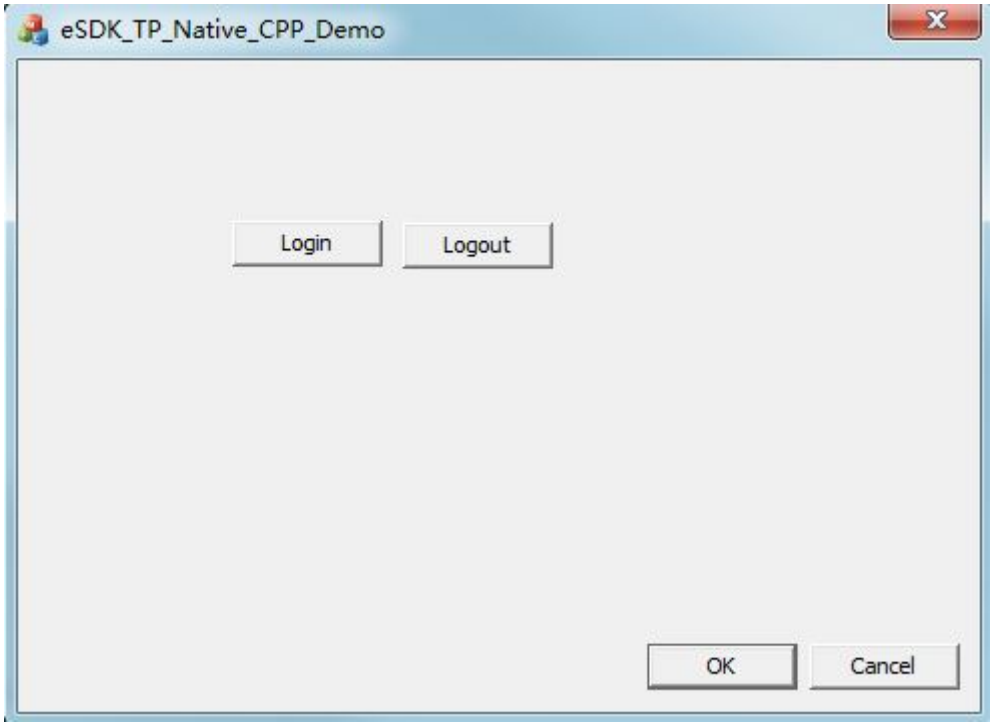


图 4-12



3. 单击“Login”按钮，页面显示“Login successful.”，表示用户登录成功。

4. 点击“Logout”按钮，页面显示“Logout successful.”，表示用户登出成功。

----结束

5 初始化配置

配置 eSDK TP 服务端

eSDK TP服务端正确配置是使用eSDK TP的前提条件，配置过程请参考**《eSDK TP V100R005C70 安装配置指南》**。

修改配置文件

请将配置文件“esdk_tp_native_professional_cs.dll.config”放在TP工程同一文件目录中，并在url的value中填写eSDK TP Server的IP地址（示例：10.45.22.67）和端口号（默认值：18543）。

```
<add key="url" value="https://${ip}:${port}/esdk/services" />
```

其中，“${ip}”表示安装eSDK TP的服务器IP地址，“${port}”表示eSDK TP提供服务的端口号。eSDK TP Server的IP地址和端口号请向管理员索取。

包含的库文件

库文件名称：TP_Native_SDK.lib

包含的头文件

头文件名称：

- TP_Native_SDK.h
- TP_Native_Types.h

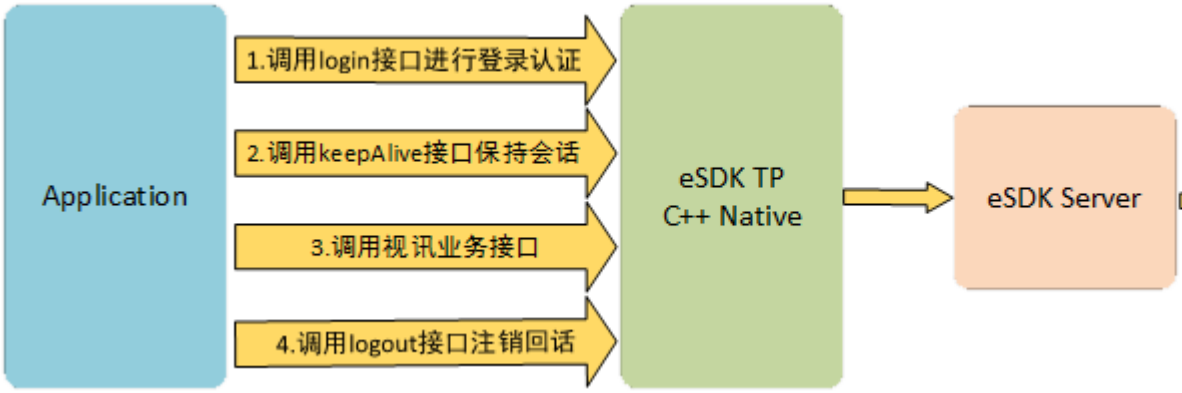
添加头文件的代码如下所示：

```
#include "TP_Native_SDK.h"
#include "TP_Native_Types.h"
```

6 典型场景开发

6.1 概述

eSDK TP提供的C++ Native SDK接口调用基本遵循如下流程：



1. 第三方应用程序（Application）调用C++ Native的**login**接口进行用户登录认证。
2. 第三方应用程序启动一个新线程，调用C++ Native的**keepAlive**接口保持会话，建议每20秒调用一次，以保证鉴权成功的session不会超时。
3. 第三方应用程序根据业务需要调用相关的业务接口进行业务处理。
4. 业务处理完成后，调用**logout**接口断开与eSDK Server的连接。

本章主要介绍如何通过eSDK TP 提供的接口进行第三方应用开发。包含4个典型的应用场景：视讯系统接入、召开两（多）方视频会议、通过会议模板预约会议（Ad hoc会议场景）和控制视频会议。

6.2 接入视讯系统

6.2.1 概述

视讯系统接入是您的系统集成视讯设备的必要场景（集成第一步），所有的视讯业务都必须经过登录才能调用相关的服务接口。系统接入主要包含三个步骤，分别是应用登录、应用保活和应用登出。

应用系统可根据实际需要选择系统接入的时机，比如用户登录到业务系统（视讯相关模块）之后选择连接视讯管理平台并保持连接，在用户登出业务系统之后登出视讯管理平台；也可以在业务系统启动之后即连接视讯管理平台，并一直保持连接。

本场景主要介绍如何通过调用eSDK TP提供的接口，实现视讯系统接入。

6.2.2 典型开发场景

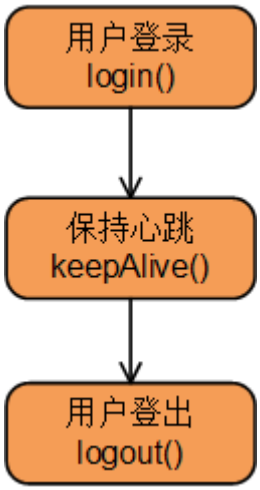
应用场景

已知用户已部署好eSDK TP Server。用户要使用eSDK TP Server，必须先登录到eSDK TP Server，并在登录期间，每隔一段时间进行保活，使用完eSDK TP Server后，用户登出eSDK TP server。

本场景需要您对接开发的接口有三个，包括**login**（用户登录）、**keepAlive**(保持心跳)、**logout**（用户登出）。

接口调用流程图如**图 1 接口调用流程图**所示。

图 6-1 接口调用流程图

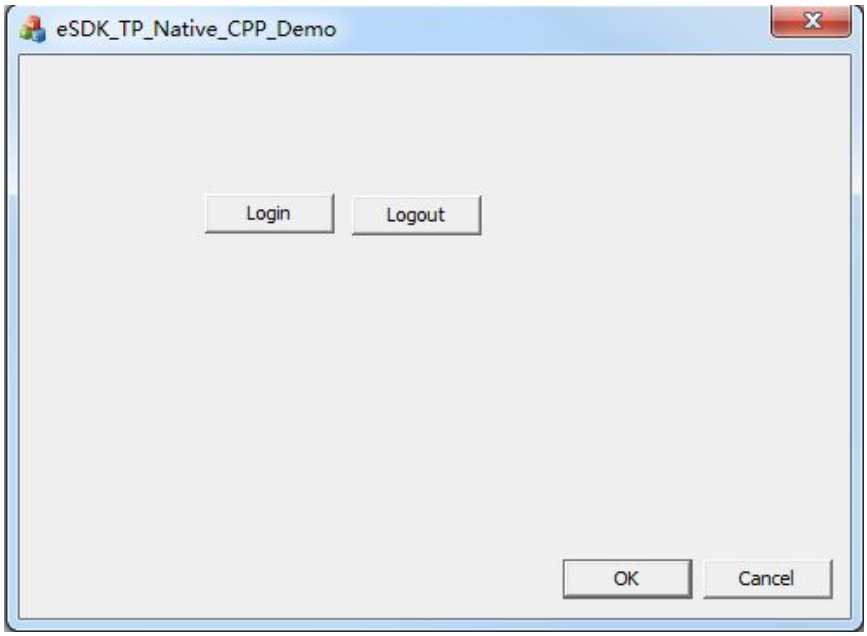


接口调用流程说明如下：

1. 第三方应用程序（Application）调用**login**接口向eSDK TP Server请求登录，并携带有用户名和密码。eSDK TP Server返回登录结果。
2. 每隔一段时间(不多余60s，建议20s)，第三方应用程序调用**keepAlive**接口向eSDK TP Server请求保活。
3. 在调用完所有业务接口后，第三方应用程序调用**logout**接口向eSDK TP Server请求登出。

接入视讯系统的Demo示意图如**图 2** 所示。

图 6-2 接入视讯系统 Demo 示意图



前提条件

- eSDK TP Server上的SMC服务已正确配置并启用，并已获取eSDK TP服务地址、端口、账号、密码。eSDK TP安装配置过程及结果验证请参考《**eSDK TP V100R005C70 安装配置指南**》。
- 已获取eSDK TP SDK软件包。

用户登录

示例如何调用**login**接口登录eSDK TP Server系统。

```
//cpp code
int ret = TP_E_RET_CODE_FAIL;
//登录
ret = login("esdk_user","Huawei@123");//登录的账号和密码请向eSDK TP Server管理员索取。

if (TP_E_RET_CODE_SUCCESS == ret){
    AfxMessageBox("登录成功");
    SetTimer(1, 19000, NULL);  //启动保活定时器
}
else{
    char result[1024]={0};
    sprintf_s(result, 1023, "登录失败;  返回码 : %d",ret);
    AfxMessageBox(result);
}
```

保持心跳

示例用户登录成功后，如何调用保活接口**keepAlive**保持登录状态。

```
//cpp code
//保活
int ret = keepAlive();
```

用户登出

示例如何调用**logout**接口登出eSDK TP Server系统。

```
//cpp code
//等待保活进程结束
StopMsgRetrieval();
//停止保活定时器
KillTimer(1);
int ret = TP_E_RET_CODE_FAIL;
//登出
ret = logout();
if (TP_E_RET_CODE_SUCCESS == ret){
    AfxMessageBox("登出成功");
}
else{
    char result[1024]={0};
    sprintf_s(result, 1023, "登出失败;  返回码 : %d",ret);
    AfxMessageBox(result);
}
```

6.3 召开两（多）方视频会议

6.3.1 概述

SMC2.0 支持以下会议召集方式：

- 在 SMC2.0 的 Web 界面上召集会议（包括从会议模板召集和新建会议）
- 通过终端呼叫 Ad hoc 会议接入号激活会议
- 通过终端呼叫统一接入号激活会议
- 通过终端呼叫统一接入号创建会议
- 通过终端主叫呼集创建会议
- 通过第三方接口召集会议

SMC2.0 作为统一的业务管理平台，支持添加 H.323、SIP（Session Initiation Protocol）类型会场。

本场景主要介绍如何通过eSDK接口召开、预约带两方或多方华为终端会场的高清视频会议。

6.3.2 典型开发场景

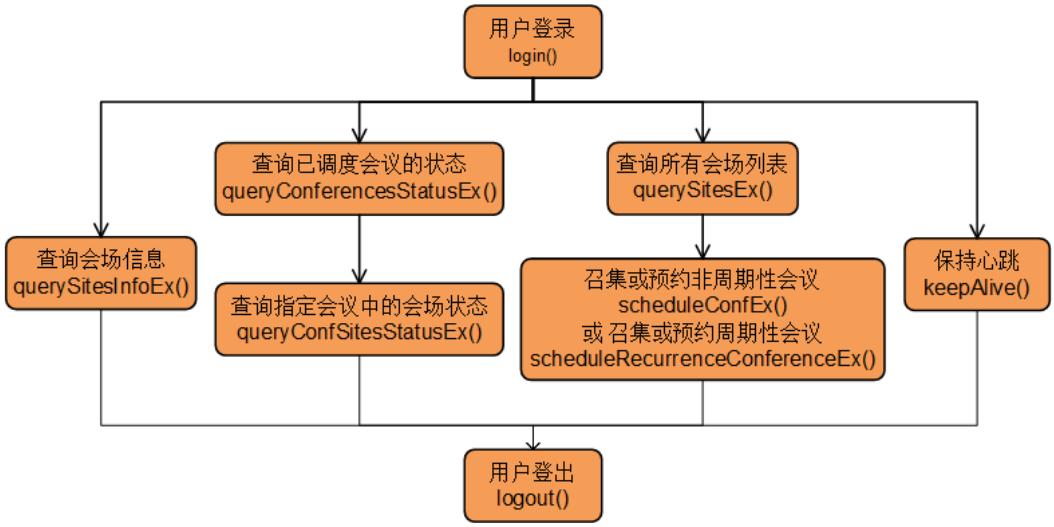
应用场景

已知客户网络部署了两个视讯会场，用户希望预定这两个视讯会场在未来某时间段召开一次视讯会议，或者希望预定每周都召开一次会议且持续一段时间；并且需要在会议开始后，查询各会场是否已出席会议。

本场景需要您对接开发的接口有五个，包括**querySitesEx**（查询所有会场列表）、**scheduleConfEx**（召集或预约非周期性会议）、**scheduleRecurrenceConferenceEx**（召集或预约周期性会议）、**queryConferencesStatusEx**（查询已调度会议的状态）、**queryConfSitesStatusEx**（查询指定会议中的会场状态）、**querySitesInfoEx**（查询会场信息）。

接口调用流程图如**图6-3**所示。

图 6-3 接口调用流程



召集会议的接口调用流程说明如下：

1. 第三方应用程序(Application)调用**querySitesEx**接口查询所有会场列表供下一步选择。

2. 用户填写会议信息，在调用**querySitesEx**接口查询的到会场列表中选择会场，然后选择会议类型（非周期性会议或周期性会议）。
3. 调用**scheduleConfEx**接口召集非周期性会议，或者调用**scheduleRecurrenceConferenceEx**接口召集周期性会议。

查询指定会议中会场状态的接口调用流程说明如下：

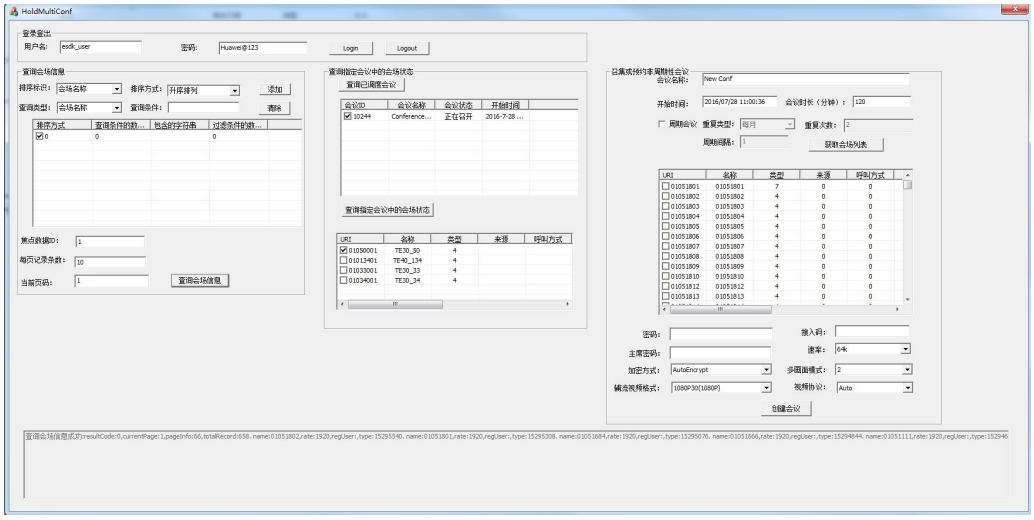
1. 第三方应用程序(Application)调用**queryConferencesStatusEx**接口查询已调度会议的状态。
2. 用户在调用**queryConferencesStatusEx**接口查询到的会议中选择会议（正在召开的会议）。
3. 调用**queryConfSitesStatusEx**接口查询选中的会议中会场的状态信息。

查询会场信息的接口调用流程说明如下：

1. 用户指定查询方式、查询关键字、排列方式以及分页信息。
2. 第三方应用程序(Application)调用**querySitesInfoEx**接口查询会场信息。

召集会议的Demo示意图如**图 2 召集会议的demo示意图**所示。

图 6-4 召集会议 Demo 示意图



前提条件

- 已获取eSDK TP SDK软件包。
- 成功登录eSDK TP Server系统，且保活正常。参考**接入视讯系统**。

查询所有会场列表

示例如何调用**querySitesEx**接口查询所有会场列表，并将查询到的会场列表显示出来。

```
//cpp code
//获得所有会场的列表
TPSDKResponseEx<list<SiteInfoEx>> siteList;
siteList = querySitesEx();

//清空m_allSitesInfo。m_allSitesInfo是一个全局变量，用来存储所查询到的会场。
vector<SiteInfoEx> m_allSitesInfo;
```

```
if (m_allSitesInfo.size()>0){
    m_allSitesInfo.clear();
}

//清空m_sitesInfoList。m_sitesInfoList是一个MFC控件，用来显示所有会场。
CListCtrl m_sitesInfoList;
m_sitesInfoList.DeleteAllItems();

//resultCode等于0，表示调用接口querySitesEx()成功
if (siteList.resultCode==0){
    int j=0;
    list<SiteInfoEx>::iterator i;
    for(i = siteList.result.begin();
        i != siteList.result.end();
        ++i) {
        //存储会场列表
        m_allSitesInfo.push_back(*i);
        //显示会场列表
        m_sitesInfoList.InsertItem(j,i->uri.c_str());
        m_sitesInfoList.SetItemText(j,1,i->name.c_str());
        CString str;
        str.Format("%d",i->type);
        m_sitesInfoList.SetItemText(j,2,str);
        str.Format("%s",i->rate.c_str());
        m_sitesInfoList.SetItemText(j,3,str);
        j++;
    }
}
```

预约非周期性会议或周期性会议

示例如何调用**scheduleConfEx**接口预约非周期性会议，如何调用**scheduleRecurrenceConferenceEx**接口创建周期性会议。用户需先获得会场列表并选择需要添加的会场，填写会议名、会议开始时间、持续时间、会议接入码、会议密码、加密方式、多画面模式、辅流视频格式、视频协议和速率，通过勾选check-box控件IDC_CHECK_RECURRENCE_CONF来切换到周期性会议。周期性会议还需要选择重复类型，填写重复次数和周期间隔。

```
//cpp code
//获得用户选择的会场
vector<int> indexTemp;
for (int i=0;i<m_sitesInfoList.GetItemCount();i++){
    if(m_sitesInfoList.GetCheck(i)) {
        indexTemp.push_back(i);
    }
}

if (indexTemp.size()>0){
    TPSDKResponseEx<ConferenceInfoEx> confInfoReturn;
    ConferenceInfoEx scheduleConf;
    UpdateData(TRUE);
    //会议名称
    scheduleConf.name = m_confName.GetBuffer();
    //会议开始时间
    COleDateTime tT;
    tT.ParseDateTime(m_beginTime);
    SYSTEMTIME st;
    tT.GetAsSystemTime(st);
    scheduleConf.beginTime = st;
    //会议时长
    int minutes = atoi(m_confTime.GetBuffer());
    int hours = minutes / 60;
    minutes = minutes - hours * 60;
    CString s;
    s.Format("P0Y0M0DT%dH%dM0.000S", hours,minutes);
    scheduleConf.duration = s.GetBuffer();
    //接入码
```

```
if ( m_accessCode.Trim().IsEmpty() == false ){
    scheduleConf.accessCode = m_accessCode.GetBuffer();
}
//密码
if (m_confPwd.GetLength()>0){
    scheduleConf.password = m_confPwd.GetBuffer();
    scheduleConf.chairmanPassword = m_chairmanPwd.GetBuffer();
}
//加密方式
scheduleConf.mediaEncryptType = m_cb_pwdtype.GetCurSel();
//多画面模式
scheduleConf.cpResouce = m_cb_multi.GetCurSel() + 1;
//辅流视频格式
scheduleConf.auxVideoFormat = m_cb_vediofm.GetCurSel();
//视频协议
scheduleConf.auxVideoProtocol = m_cb_vediopr.GetCurSel();
//速率带宽
CString strRate;
m_cb_rate.GetLBText(m_cb_rate.GetCurSel(),strRate);
scheduleConf.rate = strRate.GetBuffer();
//演示方式
scheduleConf.presentation = 0;
//录播会议是否支持直播功能
scheduleConf.isLiveBroadcast = 0;
//是否支持会议录制功能
scheduleConf.isRecording = 0;
会议状态
scheduleConf.status = 0;
//最大会场
scheduleConf.maxSitesCount = 3;
for (int j=0;j<indexTemp.size();j++){
    scheduleConf.sites.push_back(m_allSitesInfo[indexTemp[j]]);
}
m_joinConfSitesInfo = scheduleConf.sites;
if ( BST_UNCHECKED == ((CButton*)GetDlgItem(IDC_CHECK_RECURRENCE_    CONF))->GetCheck()){
    //以下代码创建普通会议
    confInfoReturn=scheduleConfEx(scheduleConf);
    if (confInfoReturn.resultCode==0){
        CString s;
        s.Format("创建会议成功；会议ID：%s",
        confInfoReturn.result.confId.c_str());
        m_showPan.SetWindowText(s.GetBuffer());
    }
    else{
        CString s;
        s.Format("创建会议失败；返回码：%d",
        confInfoReturn.resultCode);
        m_showPan.SetWindowText(s.GetBuffer());
    }
}
else{
    //以下代码是创建周期会议
    RecurrenceConfInfoEx recurrenceConf;
    recurrenceConf.base_info = scheduleConf;
    //间隔周期单位,每天/每周/每月
    recurrenceConf.frequency =
    m_cmbRecurrenceConfFrequency.GetCurSel();
    CString cstrTemp;
    m_editRecurrenceConfInterval.GetWindowText(cstrTemp);
    //间隔周期
    recurrenceConf.interval = atoi(cstrTemp.GetBuffer());
    recurrenceConf.weekDays.push_back(1);
    recurrenceConf.weekDay = 0;recurrenceConf.monthDay = 0;
    m_editRecurrenceConfCount.GetWindowText(cstrTemp);
    recurrenceConf.count = atoi(cstrTemp.GetBuffer());
    TPSDKResponseEx<RecurrenceConfInfoEx> response =
    scheduleRecurrenceConferenceEx( recurrenceConf );
    if ( 0 == response.resultCode ){
        CString s;
```

```
            s.Format("创建会议成功；会议ID：%s",
            response.result.base_info.confId.c_str());
            m_showPan.SetWindowText(s.GetBuffer());
        }
        else{
            CString s;
            s.Format("创建会议失败；返回码：%d",response.resultCode);
            m_showPan.SetWindowText(s.GetBuffer());
        }
    }
}
else{
    m_showPan.SetWindowText( _T("请先勾选要参与会议的会场"));
}
```

获取会议列表

示例演示如何调用**queryConferencesStatusEx**接口获取所有已预约的会议，并将获取的会议列表显示出来。

```
//cpp code
//清空m_listConference，m_listConference是一个MFC控件，用来显示已预订的会议。
CListCtrl m_listConference;
m_listConference.DeleteAllItems();

//清空m_conferenceList，m_conferenceList用来存储查询到的会议列表。
vector<ConferenceStatusEx> m_conferenceList;
m_conferenceList.clear();
list<std::string> confIds;
TPSDKResponseEx<list<ConferenceStatusEx>> returnInfo;
//当queryConferencesStatusEx的参数列表为空时，表示查询所有会议
returnInfo = queryConferencesStatusEx(confIds);
//resultCode等于0，表示queryConferencesStatusEx()执行成功
if ( returnInfo.resultCode == 0 ){
    int i = 0;
    for ( list<ConferenceStatusEx>::iterator
          iter = returnInfo.result.begin();
          iter != returnInfo.result.end();
          iter++ ){
        //保存会议
        m_conferenceList.push_back(*iter);
        //以下代码在MFC界面中显示会议列表及会议信息
        m_listConference.InsertItem(i,iter->id.c_str());
        m_listConference.SetItemText(i, 1, iter->name.c_str());
        CString cstrTemp;
        if ( 2 == iter->status ){
            cstrTemp=_T("待召开");
        }
        if ( 3 ==  iter->status ){
            cstrTemp=_T("正在召开");
        }
        m_listConference.SetItemText(i, 2, cstrTemp);
        cstrTemp.Format( "%d-%d-%d %d:%d:%d", iter->beginTime.wYear,
        iter->beginTime.wMonth, iter->beginTime.wDay,
        iter->beginTime.wHour, iter->beginTime.wMinute,
        iter->beginTime.wSecond);
        m_listConference.SetItemText(i, 3, cstrTemp);
    }
    m_showPan.SetWindowText(_T("查询已调度会议的状态成功"));
}
else{
    CString s;
    s.Format("查询已调度会议的状态失败；返回码：%d",returnInfo.resultCode);
    m_showPan.SetWindowText(s);
}
```

查询已调度会议的会场状态

示例演示如何调用**queryConfSitesStatusEx**接口查询指定会议中会场的状态，并将会场的状态列表显示出来。

```
//cpp code
//获得已选择的会场
ConferenceStatusEx beChoosedConf = GetChoosedConferencesItem();

if ( beChoosedConf.id != "" )
{
    list<std::string> siteUris;
    //siteUris列表为空表示查询所有会议中所有会场的状态
    TPSDKResponseEx<list<SiteStatusEx> > response = queryConfSitesStatusEx( beChoosedConf.id,
siteUris );
    if ( 0 == response.resultCode )
    {
        //清空m_listSitesInConf。m_listSitesInConf是MFC控件，用于显示会场。
        CListCtrl m_listSitesInConf;
        m_listSitesInConf.DeleteAllItems();
        for ( list<SiteStatusEx>::iterator iter = response.result.begin(); iter !=
response.result.end(); ++iter )
        {
            int i = m_listSitesInConf.GetItemCount();
            m_listSitesInConf.InsertItem( i, iter->uri.c_str() );
            m_listSitesInConf.SetItemText( i, 1, iter->name.c_str() );
            CString cstrTemp;
            cstrTemp.Format( "%d", iter->type );
            m_listSitesInConf.SetItemText( i, 2, cstrTemp );//类型
            if ( iter->status == 2 )
            {
                cstrTemp.Format( "会议中" );
            }else  {
                cstrTemp.Format( "未入会" );
            }
            m_listSitesInConf.SetItemText( i, 3, cstrTemp );//会场状态
        }
        m_showPan.SetWindowText( _T( "查询会议的会场成功" ) );
    }else  {
        CString cstrShowMsg;
        cstrShowMsg.Format( "查询会议的会场失败, 失败码=%d", response.resultCode );
        m_showPan.SetWindowText( cstrShowMsg );
    }
}else  {
    m_showPan.SetWindowText( _T( "请选择要查询的会议" ) );
}
```

查询会场信息

示例演示如何调用**querySitesInfoEx**接口按关键字分页查询会场信息，并将结果显示出来。

```
//cpp code
TPSDKResponseEx<QuerySitesInfoExResponse> returnInfo;
CString str;
CString sort;
CString itemIndex;
CString strValue;
CString columnIndex;
QueryConfigEx queryConfig;
SortItemEx sortItem;
StringFilterEx sf;
PageParamEx pageParam;
UpdateData(TRUE);
//排序条件组合
for(int i = 0;i<m_searchList.GetItemCount();++i){
    if(m_searchList.GetCheck(i)){
        sort = m_searchList.GetItemText(i,0);        //排序方式
```

```
        sortItem.sort = _ttoi(sort);
        itemIndex = m_searchList.GetItemText(i,1);
        sortItem.itemIndex = _ttoi(itemIndex);
        strValue = m_searchList.GetItemText(i,2);
        sf.value = strValue;
        columnIndex = m_searchList.GetItemText(i,3);
        sf.columnIndex = _ttoi(columnIndex);
        queryConfig.sortItems.push_back(sortItem);
        queryConfig.filters.push_back(sf);
    }
}
queryConfig.focusItem = _ttoi(m_searchID);
queryConfig.pageParam.numberPerPage = _ttoi(m_searchSum);
queryConfig.pageParam.currentPage = _ttoi(m_searchPage);
returnInfo= querySitesInfoEx(queryConfig);
if (returnInfo.resultCode==0){
    char result[1024] = {0};
    char tempResult[1024] = {0};
    sprintf_s(result,1023,"查询会场信息成功:resultCode:%d,
              currentPage:%d,pageInfo:%d,totalRecord:%d. ",
              returnInfo.resultCode,
              returnInfo.result.pageInfo.currentPage,
              returnInfo.result.pageInfo.totalPages,
              returnInfo.result.pageInfo.totalRecords);
    list<TerminalInfoEx>::iterator it;
    for ( it = returnInfo.result.sites.begin();
          it!=returnInfo.result.sites.end();
          ++it ){
        sprintf_s(tempResult,1023,
                  "name:%s,rate:%s,regUser:%s,type:%d. \n",
                  it->name.c_str(),it->rate.c_str(),
                  it->regUser.c_str(),it->uri.c_str(),
                  it->type);
        str = tempResult +str;
    }
    str = result + str;
    m_showPan.SetWindowTextA(str);
}
else{
    char result[1024] = {0};
    sprintf_s(result, 1023, "查询会场信息失败,返回码：%d", returnInfo.resultCode);
    str = result;
    m_showPan.SetWindowTextA(str);
}
```

6.4 通过会议模板预约会议（Ad hoc 会议场景）

6.4.1 概述

Ad hoc类型会议是一个类似虚拟会议室一样的会议。事先创建好有会议接入号的会议模板，视讯会场或语音会场只需要呼叫该接入号，并输入激活密码，该会议将自动被调度并分配资源。通过eSDK TP server可以创建Ad hoc会议模板，通过Ad hoc会议模板预约Ad hoc会议，并对Ad hoc模板进行修改、删除和查询。

本场景主要介绍如何通过调用eSDK TP Server提供的接口，创建会议Ad hoc会议模板，并激活Ad hoc会议。

6.4.2 典型开发场景

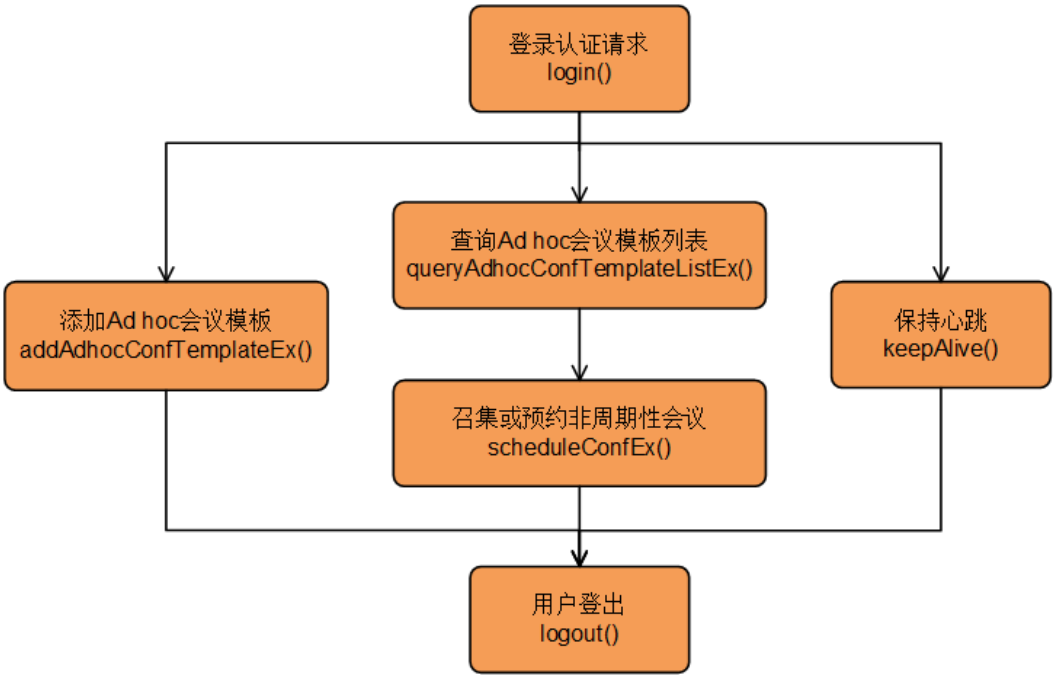
应用场景

已知客户网络部署了两个视讯会场，用户希望能够预先定义好会议模板，在有需要的时候就直接在应用系统呼叫模板ID把会议开起来，或者预约未来某时间段的会议，而不需要重新输入会议参数。

本场景需要您对接开发的接口有三个，包括**addAdhocConfTemplateEx**（添加Ad hoc会议模板）、**queryAdhocConfTemplateListEx**（查询Ad hoc会议模板列表）、**scheduleConfEx**（召集或预约周期性会议）。

预约、激活Ad hoc会议的接口调用流程如**图6-5**所示。

图 6-5 接口调用流程图



添加Ad hoc会议模板的接口调用流程说明如下：

1. 用户先填写会议模板的信息（包含会议接入号）。
2. 第三方应用程序（Application）调用**addAdhocConfTemplateEx**接口添加Ad hoc会议模板。

召集Ad hoc会议的接口调用流程说明如下：

1. 第三方应用程序（Application）调用**queryAdhocConfTemplateListEx**接口查询所有的Ad hoc会议模板。
2. 在得到的会议模板选择一个会议模板。
3. 调用**scheduleConfEx**接口把选中的会议模板转换成会议。

创建Ad hoc会议模板的Demo示意图如**图 2 创建会议模板Demo示意图**所示。

图 6-6 创建会议模板 Demo 示意图

召集Ad hoc会议的Demo示意图如**图 3 召集Ad hoc会议Demo**所示。

图 6-7 召集 Ad hoc 会议 Demo 示意图

前提条件

- 已获取eSDK TP SDK软件包。
- 成功登录eSDK TP Server系统，且保活正常。参考**接入视讯系统**。

添加 Ad hoc 会议模板

示例演示如何调用**addAdhocConfTemplateEx**接口创建Ad hoc会议模板。用户填写会议名、会议激活码、多画面资源数、流媒体加密方式、是否支持直播，获取并选择会场列表，其他信息可以不填。

```
//cpp code
std::string strOrgId = "1";
AdhocConfTemplateParamEx adhocConfTemplate;
adhocConfTemplate.adhocConfTemplateId = "0";
CString cstrTemp;
//会议名
if ( m_editAdhocConfTempName.GetWindowTextLength() > 0 ){
    m_editAdhocConfTempName.GetWindowText(cstrTemp);
    adhocConfTemplate.name = cstrTemp.Trim().GetBuffer();
}
```

```
else{
    AfxMessageBox(_T("会议名不能为空"));
    return;
}
//会议激活码
if ( m_editAdhocConfTempAccessCode.GetWindowTextLength() > 0 ){
    m_editAdhocConfTempAccessCode.GetWindowText(cstrTemp);
    adhocConfTemplate.accessCode = cstrTemp.Trim().GetBuffer();
}
else{
    AfxMessageBox(_T("会议激活码不能为空"));
    return;
}
//会议时长
if ( m_editAdhocConfTempDuration.GetWindowTextLength() > 0 ){
    m_editAdhocConfTempDuration.GetWindowText(cstrTemp);
    int minutes = atoi(cstrTemp.Trim().GetBuffer());
    int hours = minutes / 60;
    minutes = minutes - hours * 60;
    cstrTemp.Format("P0Y0M0DT%dH%dM0.000S", hours,minutes);
    adhocConfTemplate.duration = cstrTemp.GetBuffer();
}
//计费码
if ( m_editAdhocConfTempBillCode.GetWindowTextLength() > 0 ){
    m_editAdhocConfTempBillCode.GetWindowText(cstrTemp);
    adhocConfTemplate.billCode = cstrTemp.Trim().GetBuffer();
}
//密码
if (m_editAdhocConfTempPassword.GetWindowTextLength() > 0){
    m_editAdhocConfTempPassword.GetWindowText(cstrTemp);
    adhocConfTemplate.password = cstrTemp.Trim().GetBuffer();
}
//多画面资源数
adhocConfTemplate.cpResouce
    = m_cbbAdhocConfTempCpResouce.GetCurSel() + 1;
//带宽
m_cbbAdhocConfTempRate.GetLBText( m_cbbAdhocConfTempRate.GetCurSel(), cstrTemp );
adhocConfTemplate.rate = cstrTemp.Trim().GetBuffer();
//流媒体加密方式
adhocConfTemplate.mediaEncryptType
    = m_cbbAdhocConfTempMediaEncryptType.GetCurSel();
//支持直播
adhocConfTemplate.isLiveBroadcast
    = m_cbbAdhocConfTempIsLiveBroadcast.GetCurSel();
//支持录播
adhocConfTemplate.isRecording
    = m_cbbAdhocConfTempIsRecording.GetCurSel();
//最大与会方数
if ( m_editAdhocConfTempMaxSitesCount.GetWindowTextLength() > 0 ){
    m_editAdhocConfTempMaxSitesCount.GetWindowText(cstrTemp);
    adhocConfTemplate.maxSitesCount = atoi(cstrTemp.Trim());
}
//会场列表
vector<int> indexTemp;
for (int i=0; i<m_listAllSites.GetItemCount(); i++){
    if(m_listAllSites.GetCheck(i)) {
        indexTemp.push_back(i);
    }
}
for ( unsigned int j = 0; j<indexTemp.size(); j++ ){
    adhocConfTemplate.sites.push_back(m_allSitesInfo[indexTemp[j]]);
}
//主席会场
if ( m_editAdhocConfTempMainSiteUri.GetWindowTextLength() > 0 ){
    m_editAdhocConfTempMainSiteUri.GetWindowText(cstrTemp);
    adhocConfTemplate.mainSiteUri = cstrTemp.Trim().GetBuffer();
}
//主席密码
if ( m_editAdhocConfTempChairmanPassword.GetWindowTextLength() > 0 ){
```

```
    m_editAdhocConfTempChairmanPassword.GetWindowText(cstrTemp);
    adhocConfTemplate.chairmanPassword = cstrTemp.Trim().GetBuffer();
}
//演示方式
adhocConfTemplate.presentation
    = m_cbbAdhocConfTempPresentation.GetCurSel();
//辅流视频格式
adhocConfTemplate.presentationVideo.videoFormat
    = m_cbbAdhocConfTempVideoFormat.GetCurSel();
//辅流视频协议
adhocConfTemplate.presentationVideo.videoProtocol
    = m_cbbAdhocConfTempVideoProtocol.GetCurSel();
//邮箱
if ( m_editAdhocConfTempEmail.GetWindowTextLength() > 0 ){
    m_editAdhocConfTempEmail.GetWindowText(cstrTemp);
    adhocConfTemplate.notice.email = cstrTemp.Trim().GetBuffer();
}
//通知类容
if ( m_editAdhocConfTempContent.GetWindowTextLength() > 0 ){
    m_editAdhocConfTempContent.GetWindowText(cstrTemp);
    adhocConfTemplate.notice.content = cstrTemp.Trim().GetBuffer();
}
//电话
if ( m_editAdhocConfTempTelephone.GetWindowTextLength() > 0 ){
    m_editAdhocConfTempTelephone.GetWindowText(cstrTemp);
    adhocConfTemplate.notice.telephone = cstrTemp.Trim().GetBuffer();
}
//注册状态
adhocConfTemplate.state = m_cbbAdhocConfTempState.GetCurSel();
//匿名级联通道数
if ( m_editAdhocConfTempReservedCsdPorts.GetWindowTextLength() > 0 ){
    m_editAdhocConfTempReservedCsdPorts.GetWindowText(cstrTemp);
    adhocConfTemplate.reservedCsdPorts = atoi(cstrTemp.Trim());
}
TPSDKResponseEx<std::string> response
    = addAdhocConfTemplateEx( strOrgId, adhocConfTemplate );
if ( 0 == response.resultCode ){
    CString cstrMsg;
    cstrMsg.Format("创建会议模板成功；模板ID：%s",
                   response.result.c_str());
    AfxMessageBox(cstrMsg);
}
else{
    CString cstrErrMsg;
    cstrErrMsg.Format("创建会议模板失败；返回码：%d",response.resultCode);
    AfxMessageBox(cstrErrMsg);
}
```

查询 Ad hoc 会议模板列表

示例演示如何调用**queryAdhocConfTemplateListEx**接口查询Ad hoc会议模板列表，并加以显示。

```
//cpp code
m_listAllAdhocConfTemp.DeleteAllItems();
m_listAllAdhocConferencesTemplate.clear();
m_choosedAdhocConfTempItem = -1;
int iCurrentPage = 1;  //从第一页开始查询
bool bContinueLoop = true;
do {
    QueryConfigEx queryConfig;
    SortItemEx sortItemEx;
    sortItemEx.itemIndex = 4;//按会议模板的名称排序
    sortItemEx.sort = 0;
    queryConfig.sortItems.push_back(sortItemEx);
    queryConfig.focusItem = 1;
    queryConfig.pageParam.currentPage = iCurrentPage;
    queryConfig.pageParam.numberPerPage = 10;
```

```
    StringFilterEx stringFilterEx;
    stringFilterEx.columnIndex = 4;
    stringFilterEx.value = "";
    queryConfig.filters.push_back(stringFilterEx);
    TPSDKResponseWithPageEx<list<AdhocConfTemplateParamEx>> response;
    //查询Ad hoc会议模板列表
    response = queryAdhocConfTemplateListEx( queryConfig );
    if ( 0 == response.resultCode ){
        //保存查询的结果
        m_listAllAdhocConferencesTemplate.splice(
            m_listAllAdhocConferencesTemplate.end(),
            response.result);
        if ( iCurrentPage >= response.pagesInfo.totalPages ){
            //当前页码是最后一页，跳出循环
            bContinueLoop = false;
        }
        else{
            //若当前页码不是最后一页，当前页码自加1，向后翻页
            iCurrentPage++;
        }
    }
    else{
        CString cstrMsg;
        cstrMsg.Format("查询Adhoc会议模板错误，返回码：%d",
                       response.resultCode);
        AfxMessageBox(cstrMsg);
        bContinueLoop = false;
        return;
    }
} while ( true == bContinueLoop);
```

激活 Ad hoc 会议模板

示例演示如何调用**scheduleConfEx**接口激活Ad hoc会议模板。

```
//cpp code
//获得被选中的模板的迭代器
list<AdhocConfTemplateParamEx>::iterator iter
        = getChoosedAdhocConfTemplate();
if ( iter != m_listAllAdhocConferencesTemplate.end() ){
    ConferenceInfoEx scheduleConf;
    scheduleConf.name = iter->name;
    CTime m_time;
    m_time=CTime::GetCurrentTime(); //获取当前时间
    m_time.GetAsSystemTime(scheduleConf.beginTime);
    scheduleConf.duration = iter->duration;
    scheduleConf.accessCode = iter->accessCode;
    scheduleConf.password = iter->password;
    scheduleConf.mediaEncryptType = iter->mediaEncryptType;
    scheduleConf.auxVideoFormat=iter->presentationVideo.videoFormat;
    scheduleConf.auxVideoProtocol
        = iter->presentationVideo.videoProtocol;
    scheduleConf.cpResouce = iter->cpResouce;
    scheduleConf.rate = iter->rate;
    scheduleConf.isRecording = iter->isRecording;
    scheduleConf.isLiveBroadcast = iter->isLiveBroadcast;
    scheduleConf.presentation = iter->presentation;
    scheduleConf.chairmanPassword = iter->chairmanPassword;
    scheduleConf.billCode = iter->billCode;
    scheduleConf.mainSiteUri = iter->mainSiteUri;
    scheduleConf.conferenceNotice = iter->notice;
    std::copy(iter->sites.begin(),
              iter->sites.end(),
              std::back_inserter(scheduleConf.sites));
    scheduleConf.maxSitesCount = iter->maxSitesCount;
    //预约会议
    TPSDKResponseEx<ConferenceInfoEx> result
        = scheduleConfEx( scheduleConf );
    if ( 0 == result.resultCode ){
```

```
        CString cstrMsg;
        cstrMsg.Format("激活会议成功，会议ID：%s",
                       result.result.confId.c_str());
        AfxMessageBox(cstrMsg);
    }
    else{
        CString cstrMsg;
        cstrMsg.Format("激活会议失败，错误码：%d", result.resultCode);
        AfxMessageBox(cstrMsg);
    }
}
else{
    AfxMessageBox("请先选择会议");
}
```

6.5 控制视频会议

6.5.1 概述

通过eSDK TP Server可以对正在召开的会议中的会场进行控制。例如指定会议中的会场互相观看；申请或释放会议主席；指定会场静音、闭音；设置会议的多画面，广播多画面；设置会场的音量；打开或关闭会场中的视频；锁定或解锁演示者令牌。会场能被控制的前提条件是会场必须要正在会议中，被邀请但没有出席会议的会场，不能被控制。

本场景主要介绍如何通过eSDK TP server提供的接口，指定出席会议的会场互相观看；设置会议多画面并进行广播，使每个会场都观看这个多画面。

6.5.2 典型开发场景

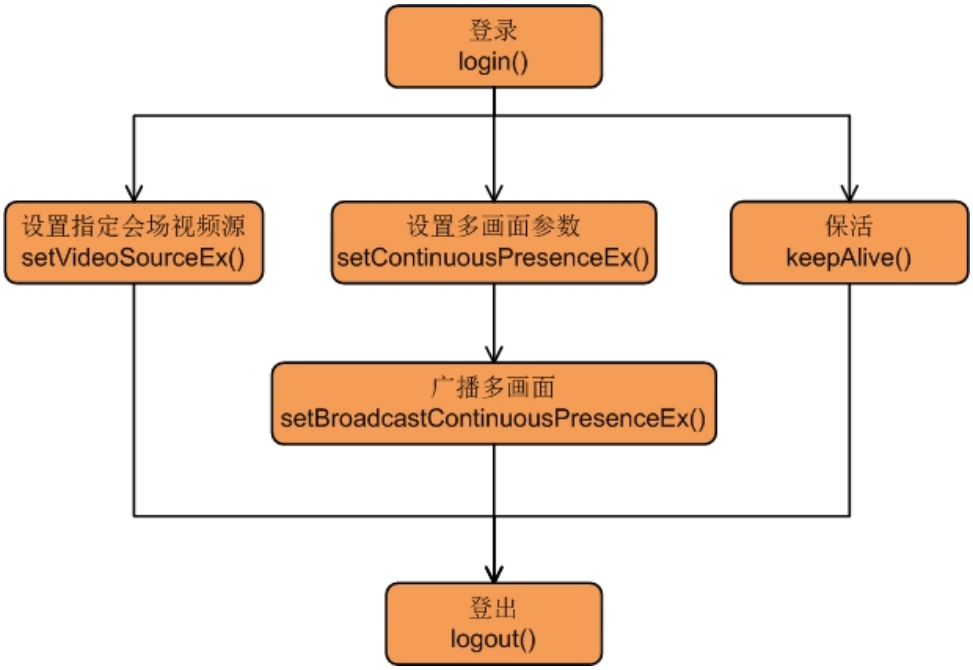
应用场景

已知有一个正在召开的视频会议中，有2个正在参加会议的会场。现在需要指定其中一个会场观看另一个会场的视频，指定会场观看视频结束后，设置所有会场观看相同的会议多画面视频。

本场景需要您对接开发的接口有三个，包括**setVideoSourceEx**（设置指定会场视频源）、**setContinuousPresenceEx**（设置多画面参数）、**setBroadcastContinuousPresenceEx**（设置广播多画面）。

图 1 接口调用流程为设置指定会场的视频源、设置多画面、广播多画面的流程图，说明了完整的业务场景。

图 6-8 接口调用流程



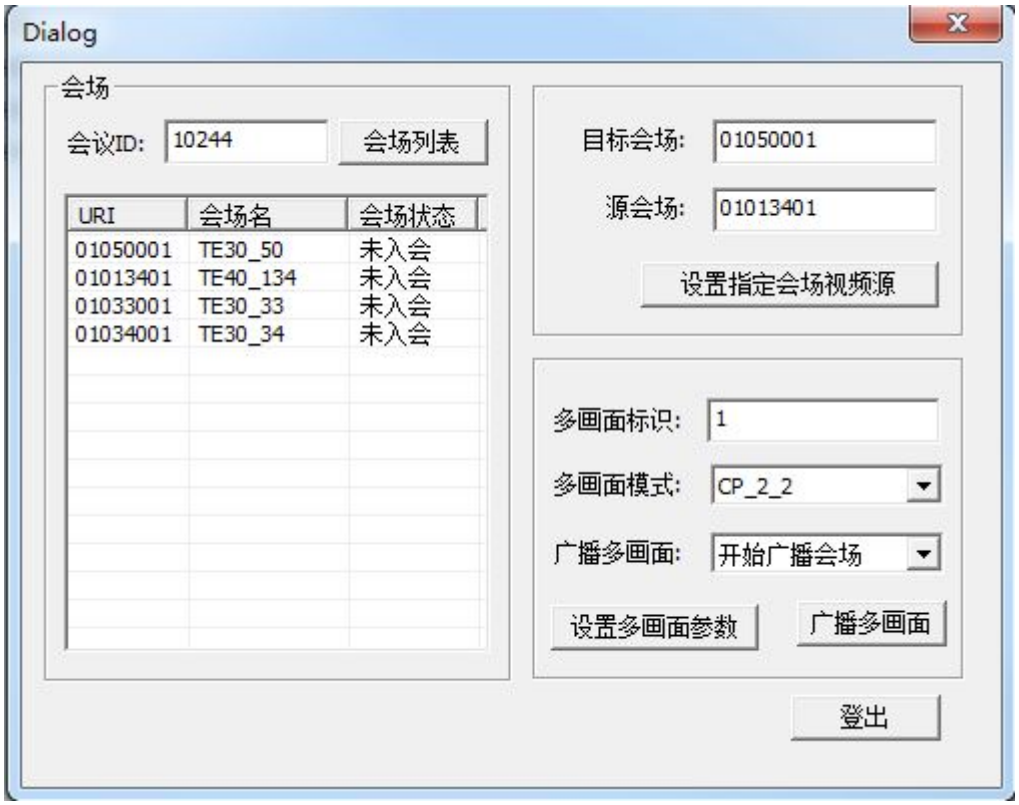
指定会场视频源的接口调用流程为：第三方应用程序（Application）调用**setVideoSourceEx**接口指定会场观看另一个会场的视频。

广播多画面的接口调用流程说明如下：

1. 第三方应用程序（Application）调用**setContinuousPresenceEx**接口设置多画面参数。
2. 调用**setBroadcastContinuousPresenceEx**接口广播多画面。

控制视频会议的Demo示意图如**图 2 控制视频会议的Demo示意图**所示。

图 6-9 控制视频会议 Demo 示意图



前提条件

- 已获取eSDK TP SDK软件包。
- 成功登录eSDK TP Server系统，且保活正常。参考**接入视讯系统**。
- 会议正常召开。

设置指定会场视频源

示例演示如何调用**setVideoSourceEx**接口设置指定会场观看指定会场视频。

```
//cpp code
CString cstrTemp;
m_editConfId.GetWindowText(cstrTemp);
std::string confID = cstrTemp.GetBuffer();
m_editSiteUri.GetWindowText(cstrTemp);
//sitUri 目标会场
std::string sitUri = cstrTemp.GetBuffer();
m_editVideoSourceUri.GetWindowText(cstrTemp);
//videoSourceUri 视频源所在会场
std::string videoSourceUri = cstrTemp.GetBuffer();
//指定会场sitUri观看会场videoSourceUri的视频
int resultCode= setVideoSourceEx(confID, sitUri, videoSourceUri, 0);
if (resultCode==0){
    AfxMessageBox("设置指定会场的视频源成功");
}
else{
    char result[1024] = {0};
    sprintf_s(result,1023,"设置指定会场的视频源失败；返回码 : %d",
              resultCode);
    AfxMessageBox(result);
}
```

设置多画面参数

示例演示如何调用**setContinuousPresenceEx**接口设置多画面参数。

```
//cpp code
CString cstrTemp;
m_editConfId.GetWindowText(cstrTemp);
std::string confId = cstrTemp.GetBuffer();
m_editTarget.GetWindowText(cstrTemp);
//target 多画面标识，当为空字符串时，所有会场观看相同的多画面
std::string target = cstrTemp.GetBuffer();
//presenceMode 多画面模式
int presenceMode = m_cbbPresenceMode.GetCurSel();
//获得会议中的会场列表
list<SiteStatusEx> sitesList = getConfAllSitesStatusEx( confId );
list<std::string> subPics;
for ( list<SiteStatusEx>::iterator iter = sitesList.begin();
      iter != sitesList.end();
      ++iter ){
    subPics.push_back(iter->uri);
}
//设置多画面参数
int iRet = setContinuousPresenceEx(confId, target,
                                   presenceMode, subPics);
if ( 0 == iRet ){
    AfxMessageBox(_T("设置多画面参数成功"));
}
else{
    CString cstrShowMsg;
    cstrShowMsg.Format("查询会议的会场失败，失败码=%d", iRet);
    AfxMessageBox(cstrShowMsg);
}
```

广播多画面

示例演示如何调用**setBroadcastContinuousPresenceEx**接口广播多画面。

```
//cpp code
CString cstrTemp;
m_editConfId.GetWindowText(cstrTemp);
std::string confID = cstrTemp.GetBuffer();
int isBroadCast = m_cbbIsBroadcast.GetCurSel();
//广播多画面。isBroadCast=0，开始广播；isBroadCast=1，停止广播
int resultCode = setBroadcastContinuousPresenceEx(confID,
                                                  isBroadCast);
if (resultCode==0){
    if ( 0 == isBroadCast ){
        AfxMessageBox("设置广播多画面成功");
    }
    else if ( 1 == isBroadCast ){
        AfxMessageBox("停止广播多画面成功");
    }
}
else{
    char result[1024] = {0};
    if ( 0 == isBroadCast ){
        sprintf_s(result,1023,"设置广播多画面失败；返回码：%d",
                  resultCode);
    }
    else if ( 1 == isBroadCast ){
        sprintf_s(result,1023,"停止广播多画面失败；返回码：%d",
                  resultCode);
    }
    AfxMessageBox(result);
}
```

7 问题定位

7.1 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；否则表示失败，该返回值即为错误码。

日志获取

日志路径：

eSDK TP的接口返回值可以通过查看接口日志文件**eSDK-SMC-API-Windows-Cpp.run.log**获取。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的“log/”目录下。

获取方法：

接口日志文件中记录了每个接口调用的时间、接口入参以及调用结果。

以用户登录接口login()为例：

```
2016-07-26 19:09:46 067| CRIT|[8744]==============log start============
2016-07-26 19:09:46 068| INFO|[8744]Upload timer starts. | IP:esdk-log.huawei.com
2016-07-26 19:09:46 068| INFO|[8744]|SDK Compile Time:Jun 12 2016 17:55:46
2016-07-26 19:09:46 902|DEBUG|[8744]|Enter login
2016-07-26 19:09:46 902|DEBUG|[8744]|Enter CTPCommon::Login username:esdk_user
2016-07-26 19:09:46 903|DEBUG|[8744]|Enter CTPCommon::LoginRequest username:esdk_user
2016-07-26 19:09:46 903|DEBUG|[8744]|Enter XMLProcess::SetXml_LoginRequest
2016-07-26 19:09:46 904|DEBUG|[8744]|Leave XMLProcess::SetXml_LoginRequest
2016-07-26 19:09:47 476|DEBUG|[8744]|Enter XMLProcess::GetRSPXml_LoginRequest
2016-07-26 19:09:47 476|DEBUG|[8744]|Enter XMLProcess::GetRSPXML_ResultCode
2016-07-26 19:09:47 476|DEBUG|[8744]|Leave XMLProcess::GetRSPXML_ResultCode
2016-07-26 19:09:47 476|DEBUG|[8744]|Leave XMLProcess::GetRSPXml_LoginRequest
2016-07-26 19:09:47 476|DEBUG|[8744]|Leave CTPCommon::LoginRequest
2016-07-26 19:09:47 476|DEBUG|[8744]|Enter CTPCommon::Authenticate
2016-07-26 19:09:47 477|DEBUG|[8744]|Enter XMLProcess::SetXml_Authenticate
2016-07-26 19:09:47 477|DEBUG|[8744]|Leave XMLProcess::SetXml_Authenticate
2016-07-26 19:10:08 527|DEBUG|[8744]|Enter XMLProcess::GetRSPXML_ResultCode
2016-07-26 19:10:08 527|DEBUG|[8744]|Leave XMLProcess::GetRSPXML_ResultCode
2016-07-26 19:10:08 527|ERROR|[8744]|Invalid response
2016-07-26 19:10:08 527|DEBUG|[8744]|Leave CTPCommon::Authenticate
2016-07-26 19:10:08 528|ERROR|[8744]|Request certification failed, error code[2130000011]
2016-07-26 19:10:08 528|DEBUG|[8744]|Leave CTPCommon::Login
2016-07-26 19:10:08 528|DEBUG|[8744]|Leave login
2016-07-26 19:10:11 848| CRIT|[8744]==============log end==============
```

搜索接口关键字“Login”，可以查看该接口调用信息。日志文件中的“error code[2130000011]”表示错误码是2130000011。

7.2 错误码查询方法

在接口参考文档中，列出了所有错误码信息，包括eSDK错误码和TP产品错误码，如图 1 所示。

根据调用接口返回的错误码，即可查询到相对应的错误描述。

图 7-1 错误码

错误码	错误描述	可能原因	处理步骤
0	成功	操作成功	无须处理
4	消息中缺少必选参数	Soap消息必选项为空	对照SDK接口说明，填写必选项内容
5	消息中传入的参数值非法	Soap消息的参数格式不合法	对照SDK接口说明，检查各个参数的合法性
570789910	参数错误	输入的参数错误	请参见接口文档相关章节，核对输入的参数
2130000010	SDK内部服务错误	• SDK内部转换或处理异常 • 输入参数的字节数超过系统设定长度限制值 • 必填参数置空，请检查具体参数	• 根据日志定位 • 将输入参数字节数调整为设定长度内 • 将置空的必填参数填入相应内容
2130000011	eSDK与设备间网络异常	eSDK与设备间网络异常	无
2130000012	设备连接失败	设备存在问题，导致eSDK连接不上	检查设备，查看日志确定错误原因
2130000013	设备不在平台列表中	• 设备配置文件中的deviceVersion或deviceType配置错误 • 调用的终端接口的Uri未在devices.xml中配置	请检查系统是否正常工作，并检查相关配置文件
2130000014	应用向eSDK鉴权失败	第三方应用输入的appId或password不正确	请检查appId或password是否输入正确或者消息节点是否正确
2130000015	应用未开通能力	应用功能位没有开通	查看SDK应用服务的功能位开通情况，将相应功能开通即可

通过查询接口参考文档的错误码可知，调用login接口时返回错误码“2130000011”，错误描述是“SDK内部服务错误“，可以根据可能原因及处理步骤进行排错。

7.3 日志分析

日志解析

1. 日志文件中，每一对“log start”和“log end”，都是一次程序运行的过程，如**图1**所示。

图 7-2 接口调用日志文件

```
2016-07-28 10:25:44 844| CRIT|[7956]==============log start=============
2016-07-28 10:25:44 844| INFO|[7956]Upload timer starts. | IP:esdk-log.huawei.com
2016-07-28 10:25:44 844| INFO|[7956]|SDK Compile Time:Jun 12 2016 17:55:46
2016-07-28 10:25:45 847|DEBUG|[7956]|Enter login
2016-07-28 10:25:45 847|DEBUG|[7956]|Enter CTPCommon::Login username:esdk_user
2016-07-28 10:25:45 848|DEBUG|[7956]|Enter CTPCommon::LoginRequest username:esdk_user
2016-07-28 10:25:45 848|DEBUG|[7956]|Enter XMLProcess::SetXml_LoginRequest
2016-07-28 10:25:45 849|DEBUG|[7956]|Leave XMLProcess::SetXml_LoginRequest
2016-07-28 10:25:46 370|DEBUG|[7956]|Enter XMLProcess::GetRSPXml_LoginRequest
2016-07-28 10:25:46 370|DEBUG|[7956]|Enter XMLProcess::GetRSPXML_ResultCode
2016-07-28 10:25:46 371|DEBUG|[7956]|Leave XMLProcess::GetRSPXML_ResultCode
2016-07-28 10:25:46 371|DEBUG|[7956]|Leave XMLProcess::GetRSPXml_LoginRequest
2016-07-28 10:25:46 371|DEBUG|[7956]|Leave CTPCommon::LoginRequest
2016-07-28 10:25:46 371|DEBUG|[7956]|Enter CTPCommon::Authenticate
2016-07-28 10:25:46 371|DEBUG|[7956]|Enter XMLProcess::SetXml_Authenticate
2016-07-28 10:25:46 372|DEBUG|[7956]|Leave XMLProcess::SetXml_Authenticate
2016-07-28 10:25:46 486|DEBUG|[7956]|Enter XMLProcess::GetRSPXML_ResultCode
2016-07-28 10:25:46 486|DEBUG|[7956]|Leave XMLProcess::GetRSPXML_ResultCode
2016-07-28 10:25:46 487|DEBUG|[7956]|Leave CTPCommon::Authenticate
2016-07-28 10:25:46 487|DEBUG|[7956]|Leave CTPCommon::Login
2016-07-28 10:25:46 488|DEBUG|[7956]|Leave login
2016-07-28 10:26:25 539|DEBUG|[9572]|Enter keepAlive
2016-07-28 10:26:25 540|DEBUG|[9572]|Enter CTPCommon::KeepAlive
2016-07-28 10:26:25 541|DEBUG|[9572]|Enter XMLProcess::SetXml_KeepAlive
2016-07-28 10:26:25 541|DEBUG|[9572]|Leave XMLProcess::SetXml_KeepAlive
2016-07-28 10:26:25 589|DEBUG|[9572]|Enter XMLProcess::GetRSPXML_ResultCode
2016-07-28 10:26:25 590|DEBUG|[9572]|Leave XMLProcess::GetRSPXML_ResultCode
2016-07-28 10:26:25 590|DEBUG|[9572]|Leave CTPCommon::KeepAlive
2016-07-28 10:26:25 590|DEBUG|[9572]|Leave keepAlive
2016-07-28 10:26:49 257|DEBUG|[7956]|Enter logout
2016-07-28 10:26:49 257|DEBUG|[7956]|Enter CTPCommon::Logout
2016-07-28 10:26:49 257|DEBUG|[7956]|Enter XMLProcess::SetXml_Logout
2016-07-28 10:26:49 257|DEBUG|[7956]|Leave XMLProcess::SetXml_Logout
2016-07-28 10:26:49 345|DEBUG|[7956]|Enter XMLProcess::GetRSPXML_ResultCode
2016-07-28 10:26:49 346|DEBUG|[7956]|Leave XMLProcess::GetRSPXML_ResultCode
2016-07-28 10:26:49 347|DEBUG|[7956]|Leave CTPCommon::Logout
2016-07-28 10:26:49 347|DEBUG|[7956]|Leave logout
2016-07-28 10:26:49 395| CRIT|[7956]==============log end==============
```

2. 在调用和结束调用每一个接口时，都会打印一条“Enter”和“Leave”日志。图1中“Enter login”和“Leave Login”，表示开始调用login接口和结束调用login接口。

具体分析

由下图可知，调用login接口时，第569行开始响应，第592行响应结束。

```
569 2016-07-28 16:11:55 200| CRIT|[10808]==============log start============
570 2016-07-28 16:11:55 200| INFO|[10808]Upload timer starts. | IP:esdk-log.huawei.com
571 2016-07-28 16:11:55 200| INFO|[10808]|SDK Compile Time:Jun 12 2016 17:55:46
572 2016-07-28 16:11:58 691|DEBUG|[10808]|Enter login
573 2016-07-28 16:11:58 692|DEBUG|[10808]|Enter CTPCommon::Login username:esdk_user
574 2016-07-28 16:11:58 692|DEBUG|[10808]|Enter CTPCommon::LoginRequest username:esdk_user
575 2016-07-28 16:11:58 692|DEBUG|[10808]|Enter XMLProcess::SetXml_LoginRequest
576 2016-07-28 16:11:58 693|DEBUG|[10808]|Leave XMLProcess::SetXml_LoginRequest
577 2016-07-28 16:11:59 226|DEBUG|[10808]|Enter XMLProcess::GetRSPXml_LoginRequest
578 2016-07-28 16:11:59 226|DEBUG|[10808]|Enter XMLProcess::GetRSPXML_ResultCode
579 2016-07-28 16:11:59 227|DEBUG|[10808]|Leave XMLProcess::GetRSPXML_ResultCode
580 2016-07-28 16:11:59 227|DEBUG|[10808]|Leave XMLProcess::GetRSPXml_LoginRequest
581 2016-07-28 16:11:59 227|DEBUG|[10808]|Leave CTPCommon::LoginRequest
582 2016-07-28 16:11:59 227|DEBUG|[10808]|Enter CTPCommon::Authenticate
583 2016-07-28 16:11:59 227|DEBUG|[10808]|Enter XMLProcess::SetXml_Authenticate
584 2016-07-28 16:11:59 228|DEBUG|[10808]|Leave XMLProcess::SetXml_Authenticate
585 2016-07-28 16:11:59 246|DEBUG|[10808]|Enter XMLProcess::GetRSPXML_ResultCode
586 2016-07-28 16:11:59 247|DEBUG|[10808]|Leave XMLProcess::GetRSPXML_ResultCode
587 2016-07-28 16:11:59 247|ERROR|[10808]|Invalid response
588 2016-07-28 16:11:59 247|DEBUG|[10808]|Leave CTPCommon::Authenticate
589 2016-07-28 16:11:59 247|ERROR|[10808]|Request certification failed, error code[1344274482]
590 2016-07-28 16:11:59 247|DEBUG|[10808]|Leave CTPCommon::Login
591 2016-07-28 16:11:59 247|DEBUG|[10808]|Leave login
592 2016-07-28 16:12:00 973| CRIT|[10808]==============log end==============
```

本例是登录出错的场景，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程序出错的具体原因。

步骤1 在Debug日志文件中找到“Leave login”。本例中是第591行。

```
591 2016-07-28 16:11:59 247|DEBUG|[10808]|Leave login
```

步骤2 找到“Leave login”上方最近的一条“error code”日志。本例中是第589行。

```
586 2016-07-28 16:11:59 247|DEBUG|[10808]|Leave XMLProcess::GetRSPXML_ResultCode
587 2016-07-28 16:11:59 247|ERROR|[10808]|Invalid response
588 2016-07-28 16:11:59 247|DEBUG|[10808]|Leave CTPCommon::Authenticate
589 2016-07-28 16:11:59 247|ERROR|[10808]|Request certification failed, error code[1344274482]
590 2016-07-28 16:11:59 247|DEBUG|[10808]|Leave CTPCommon::Login
591 2016-07-28 16:11:59 247|DEBUG|[10808]|Leave login
592 2016-07-28 16:12:00 973| CRIT|[10808]==============log end==============
```

步骤3 分析日志上下文。

本例中第589行显示的是请求认证失败，由SMC返回的错误码为“1344274482”，查看接口参考文档错误码章节，可知对应的错误描述为“用户账号或密码错误”。

1344274481	无效用户。
1344274482	用户账号或密码错误。
1344274483	录播服务器已经存在。
1344274484	录播服务器不存在。
1344274485	用户账号被锁定，请联系管理员解锁。

步骤4 检查输入的账号和密码之后，发现输入的密码有误。重新输入正确密码后登录成功。

----结束

8 修订记录