|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
|  | | |
|  | | |
| 方圆ok1 | | | |
|  | eSDK TP V100R005C60开发指南01 (TE Desktop,Windows) | | 附件1-16K |
|  | |
| **文档版本** | **01** |
| **发布日期** | **2016-04-01** |
|  | |
| 华为技术有限公司 | |
|  | | |

|  |
| --- |
| 版权所有 © 华为技术有限公司2016。 保留一切权利。  非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。  商标声明  附件3-版权声明页图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](mailto:support@huawei.com) |
| 客户服务电话： | 4008302118 |

目 录

[1 前言 2](#_Toc447282023)

[2 快速入门 4](#_Toc447282024)

[3 接口调用流程 13](#_Toc447282025)

[4 场景一：点对点呼叫 14](#_Toc447282026)

[5 场景二：创建会议 21](#_Toc447282027)

[6 场景三：会控操作 24](#_Toc447282028)

[7 场景四：通讯录操作 30](#_Toc447282029)

[7.1 通讯录基本操作 30](#_Toc447282030)

[7.2 搜索企业通讯录联系人 38](#_Toc447282031)

[7.3 历史通话记录操作 40](#_Toc447282032)

[8 场景五：辅流控制 42](#_Toc447282033)

# 前言

目的

本文档主要介绍如何通过eSDK TE Desktop进行第三方应用的开发，主要包括开发快速入门、接口调用流程和集成场景实现。

配套产品版本

与本文档相配套的产品版本如下所示：

|  |  |
| --- | --- |
| 产品名称 | 版本 |
| eSDK TE Desktop | V100R005C60 |
| TE Desktop | V500R002C00SPC100B012 |
| HUAWEI SMC2.0 | V500R002C00 |
| HUAWEI MCU | VP9660 V500R002C00B061 |

读者对象

本文档（本指南）主要适用于以下对象：

* ISV软件开发工程师
* ISV架构师

符号约定

在本文档中可能出现下列标志，它们所代表的含义如下。

| 符号 | 说明 |
| --- | --- |
| 3.png | 用于警示紧急的危险情形，若不避免，将会导致人员死亡或严重的人身伤害。 |
| 2.png | 用于警示潜在的危险情形，若不避免，可能会导致人员死亡或严重的人身伤害。 |
| 备份.png | 用于警示潜在的危险情形，若不避免，可能会导致中度或轻微的人身伤害。 |
| 0.png | 用于传递设备或环境安全警示信息，若不避免，可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。  “注意”不涉及人身伤害。 |
| 说明 | 用于突出重要/关键信息、最佳实践和小窍门等。  “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。 |

修订记录

| 日期 | 修订版本 | 描述 |
| --- | --- | --- |
| 2016-04-01 | V1.5.60 | V100R005C60版本文档正式发布 |

# 快速入门

准备开发环境

在进行eSDK TE Desktop二次开发之前，应该准备好相应开发工具，工具名称和版本要求如下所示：

|  |  |
| --- | --- |
| 工具名称 | 版本要求 |
| 操作系统 | Windows 7专业版32/64位 |
| Microsoft Visual Studio | Visual Studio 2008 SP/2010专业版 |
| eSDK TE Desktop | V100R005C60 |

下载eSDK TE Desktop开发包

点击<http://developer.huawei.com/ilink/esdk/download/HW_474854>下载eSDK TE SDK开发包。其中，software文件夹下的压缩包eSDK\_TP\_TE\_SDK\_V1.5.60解压缩后得到以下目录结构：



在bin和lib文件夹中分别包含以下内容：

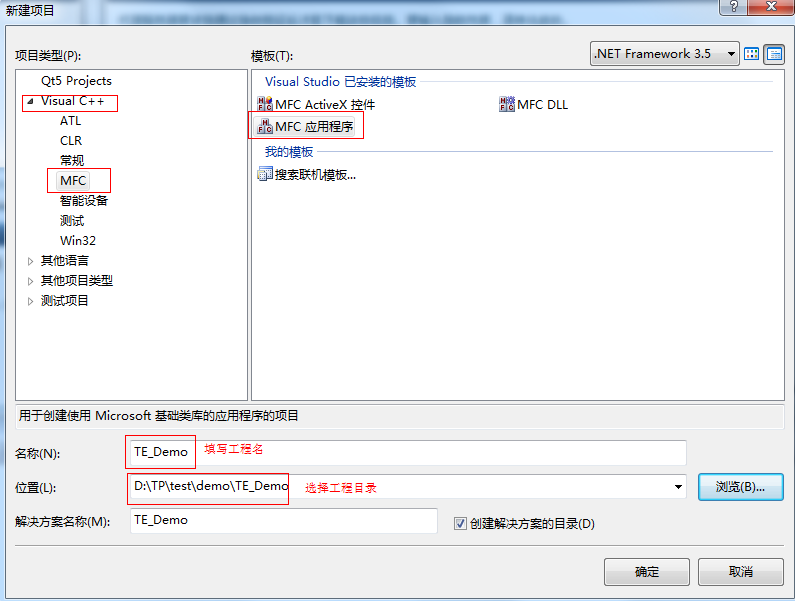
1

使用eSDK TE Desktop开发包

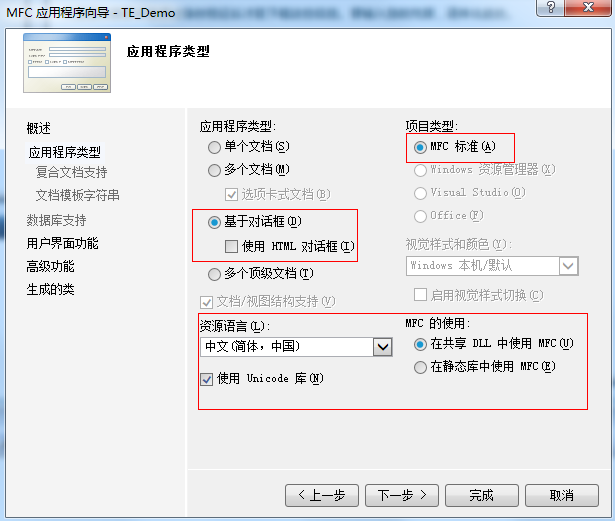
以用户登录为例，介绍如何快速在工程中引入eSDK库文件，并进行二次开发。

通过Microsoft Visual Studio 2008新建MFC工程，在工程中导入eSDK TE Desktop开发包。具体步骤如下：

（1）在Microsoft Visual Studio 2008开发工具中，单击“文件>新建>项目”，弹出新建项目对话框，完成后单击“确定”。



（2）之后的对话框，默认单击“下一步”，直到遇到下图对话框，配置如图，完成后单击“下一步”。

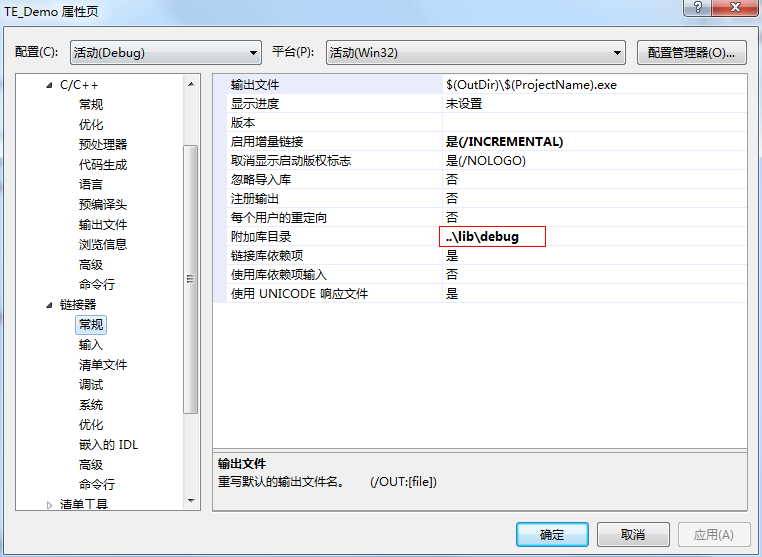


（3）之后对话框全部默认单击“下一步”，直到最后的对话框单击“完成”即可。

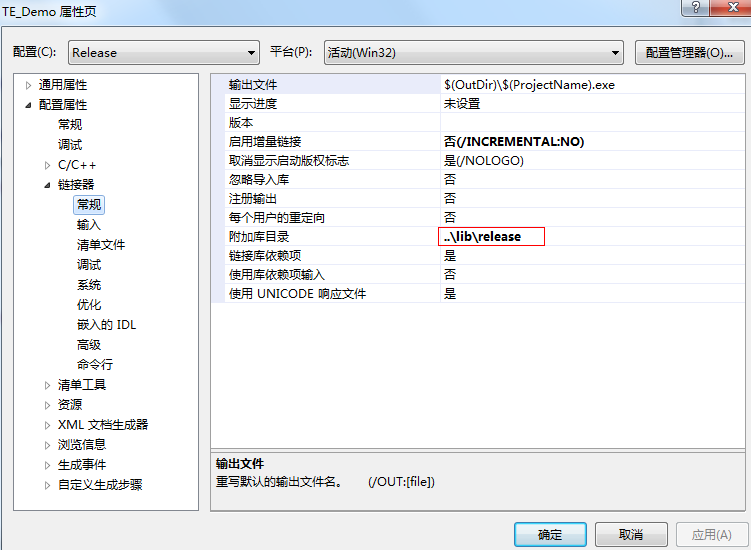
image013

eSDK提供debug和release版本的库文件，在配置工程时，选择其中一个配置即可。

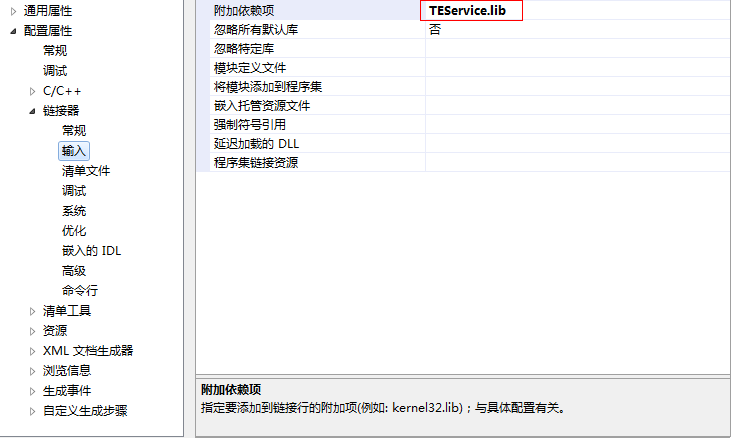
Debug下附加库目录配置示例如下：



Release下附加库目录配置示例如下：



Debug或Release下的引入库配置示例如下：



初始化TE SDK，设置TE Desktop回调函数。代码如下：

CTE\_DemoDlg::CTE\_DemoDlg(CWnd\* pParent /\*=NULL\*/): CDialog(CTE\_DemoDlg::IDD, pParent)

{

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

//初始化TE SDK

TE\_SDK\_Init();

//设置TE Desktop回调函数

TE\_SDK\_SetEventCallBack(TeEventCallBack,this);

…

}

通过TE Desktop回调函数获取登录状态，显示登录结果。代码如下：

void CALLBACK TeEventCallBack(

TE\_INT32 iEventType, //事件类型

TE\_VOID\* pEventBuf, //事件数据

TE\_UINT32 uiBufSize, //buf长度

TE\_VOID\* pUserData //用户数据，自定义，相应回调函数被调用时返回该参数

)

{

switch(iEventType)

{

//登录事件的结果

case TE\_E\_EVT\_LOGIN\_RESULT:

{

CTE\_DemoDlg \*pWnd =NULL;

TE\_S\_LOGIN\_RESULT \*pLoginResult = NULL;

pWnd = (CTE\_DemoDlg \*)pUserData;

pLoginResult = (TE\_S\_LOGIN\_RESULT \*)pEventBuf;

if(!pWnd || !pLoginResult)

{

return;

}

::PostMessage(pWnd->GetSafeHwnd(), WM\_SIGNIN\_NOTIFY, WPARAM(pLoginResult->loginState), NULL); // 发送登录状态信息

break;

}

…

//视频呼叫的结果

case TE\_EVT\_VIDEO\_CALL\_RESULT:

{

…

}

//音频呼叫的结果

case TE\_EVT\_AUDIO\_CALL\_RESULT:

{

…

}

default:

{

break;

}

}

return;

}

在MFC工程中添加一个IP地址控件、三个编辑框控件（分别表示服务器端口、登录账号和登录密码）和一个登录按钮控件，在按钮事件中添加如下代码：

int len = 0,ret=0;

TE\_S\_LOGIN\_INFO LoginReqInfo;

BYTE field0, field1, field2, field3;

memset(&LoginReqInfo, 0, sizeof(TE\_S\_LOGIN\_INFO));

if (m\_isLogin)

{

AfxMessageBox(\_T("Already Login in."));

return;

}

UpdateData(TRUE);

//获取IP数据

m\_ipCtrlServerIP.GetAddress(field0,field1,field2,field3);

m\_CStrServerIP.Format(\_T("%d.%d.%d.%d"),field0,field1,field2,field3);

//获取用户名数据

len = m\_CStrUsername.GetLength();

strncpy(LoginReqInfo.cUserName ,m\_CStrUsername.GetBuffer(len),len);

//获取密码数据

len = m\_CStrPassword.GetLength();

strncpy(LoginReqInfo.cPWD ,m\_CStrPassword.GetBuffer(len),len);

//获取端口数据

len = m\_CStrPortNum.GetLength();

LoginReqInfo.uiPort = atoi(m\_CStrPortNum.GetBuffer(len));

len = m\_CStrServerIP.GetLength();

strncpy(LoginReqInfo.stIP,m\_CStrServerIP.GetBuffer(len),len);

strncpy(LoginReqInfo.cTransMode,"TLS",3);

//调用用户登录接口

ret = TE\_SDK\_Login(&LoginReqInfo);

if (TE\_SDK\_Success == ret)

{

//CDialog::OnOK();

}

else

{

CString msg;

msg.Format("Login Failure: %d", ret);

SetWindowText(msg);

}

处理WM\_SIGNIN\_NOTIFY消息函数。代码如下：

BEGIN\_MESSAGE\_MAP(CTE\_DemoDlg, CDialog)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

//}}AFX\_MSG\_MAP

...

ON\_MESSAGE(WM\_SIGNIN\_NOTIFY, &CTE\_DemoDlg::OnSigninNotify)

...

END\_MESSAGE\_MAP()

…

LRESULT CTE\_DemoDlg::OnSigninNotify(WPARAM wParm, LPARAM Lparam)

{

int state = (int)wParm;

m\_isLogin = 0;

if (TE\_LOGIN\_SUCCESS == state)

{

AfxMessageBox(\_T("Login success"));

m\_isLogin = 1;

}

else if (TE\_LOGIN\_SIGNOUT == state)

{

AfxMessageBox(\_T("Logout"));

}

else

{

AfxMessageBox(\_T("Login failure"));

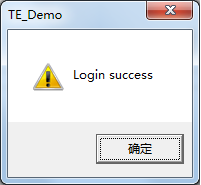
}

return 0L;

}

编译运行工程，完成登录。

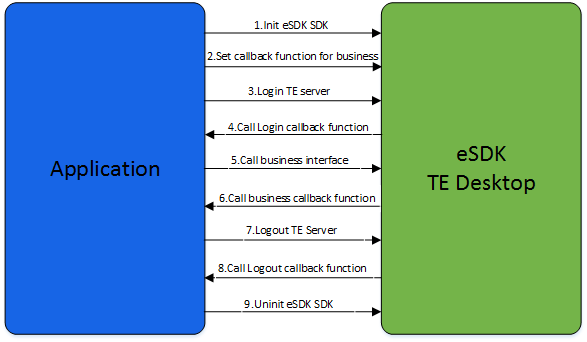
如果弹出“Login success”对话框，表示登录成功。



----结束

# 接口调用流程

接口调用流程如下所示：



在调用TE Desktop所有接口之前必须先初始化TE SDK；

设置TE Desktop回调函数，通过回调函数返回TE的状态信息，如登录成功与否，音频/视频呼叫等；

在调用所有业务接口之前，必须先完成登录；

当调用登录接口之后，TE SDK会通过之前设置的回调函数，上抛登录结果；

调用业务相关接口，如进行音频/视频通话等；

通过之前设置的回调函数，TE SDK会告诉第三方开发者当前业务的状态信息，并进行相应的流程处理；

业务结束，要先调用登出接口；

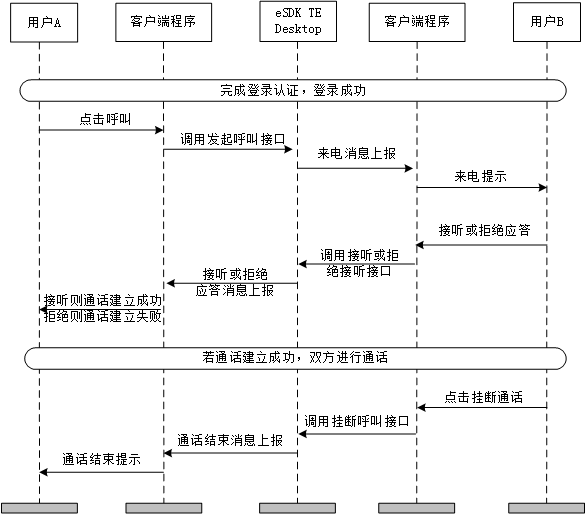
TE SDK会告知第三方开发者登出结果；

释放TE SDK资源。

# 场景一：点对点呼叫

业务流程

实现点对点呼叫（包括音频/视频通话）的时序图如下所示：



实现点对点呼叫的流程说明如下：

用户A点击呼叫，客户端程序调用TE\_MakeCall接口发起呼叫。

用户B客户端程序收到来电回调通知消息，消息ID为TE\_E\_CALL\_EVENT\_AUDIOVIDEOINCOMING。

用户A客户端程序收到TE\_E\_EVT\_CALL\_RESULT的回调消息，表示用户B正在振铃。

用户B客户端程序根据是音频来电还是视频来电判断调用TE\_AnswerAudioCall或TE\_AnswerVideoCall接口接听来电。

用户A客户端程序收到TE\_E\_CALL\_EVENT\_ANSWERRSP的回调消息，表示用户B已经接听来电。

用户A程序界面提示呼叫接通，开始计算呼叫接通时间。

通话结束时，用户B主动点击挂断，客户端程序调用TE\_DropCall接口结束呼叫。

TE SDK向用户A客户端程序上报通话结束消息。

用户A客户端程序接收呼叫结束的回调消息TE\_E\_CALL\_EVENT\_CLOSE。

用户A客户端界面提示呼叫已挂断。

image013

用户B同样可以执行发起呼叫操作，用户A同样可以执行挂断呼叫操作。

实现说明

用户A点击呼叫，客户端程序调用TE\_MakeCall发起呼叫，传入被叫的绑定号码。代码如下：

int len = 0;

char cCallNum[512]={0};

if (!m\_isLogin)

{

AfxMessageBox(\_T("Please Login in."));

return;

}

UpdateData(TRUE);

len = m\_CStrCallNum.GetLength();

strncpy(cCallNum, m\_CStrCallNum.GetBuffer(len), 511);

if(m\_btnVideo.GetCheck())

{

//调用视频呼叫接口

TE\_MakeCall(cCallNum, TE\_E\_CALL\_TYPE\_IPVIDEO);

}

else

{

//调用音频呼叫接口

TE\_MakeCall(cCallNum, TE\_E\_CALL\_TYPE\_IPAUDIO);

}

if(!m\_incomingCall)

{

m\_incomingCall = new CTE\_IncomingCall();

m\_incomingCall->Create(IDD\_COMMINGCALL\_DIALOG);//create non-modal dialog.

}

if(!m\_incomingCall)

{

return;

}

if(m\_btnVideo.GetCheck())

{

m\_incomingCall->m\_isVideo = TRUE;

}

m\_incomingCall->m\_bIsHangup = false;

m\_incomingCall->ShowWindow(SW\_SHOWNORMAL);

m\_incomingCall->SetForegroundWindow();//create thread of specified window to foreground and activate this window.

用户B客户端程序收到来电回调通知消息，消息ID为TE\_E\_CALL\_EVENT\_AUDIOVIDEOINCOMING。代码如下：

TE\_S\_CALL\_STATE\_INFO \*param = (TE\_S\_CALL\_STATE\_INFO \*)Lparam;

if (NULL == param)

{

AfxMessageBox(\_T("Error: Lparam is Null Pointer"));

return 0L;

}

if(!m\_incomingCall)

{

m\_incomingCall = new CTE\_IncomingCall();

m\_incomingCall->Create(IDD\_COMMINGCALL\_DIALOG);

}

m\_incomingCall->m\_bIsHangup = false;

m\_incomingCall->m\_CallState.Format(\_T("Calling..."));

memcpy(&m\_incomingCall->m\_CallStateInfo, param, sizeof(TE\_S\_CALL\_STATE\_INFO));

if (m\_incomingCall->m\_CallStateInfo.enCallType == TE\_E\_CALL\_TYPE\_IPVIDEO)

{

m\_incomingCall->m\_isVideo = TRUE;

}

m\_incomingCall->ShowWindow(SW\_SHOWNORMAL);

m\_incomingCall->SetForegroundWindow();

::PostMessage(m\_incomingCall->GetSafeHwnd(), WM\_CALL\_UPDATE\_STATE, NULL, NULL);

delete param;

用户A客户端程序收到TE\_E\_EVT\_CALL\_RESULT的回调消息，表示用户B正在振铃。代码如下：

TE\_S\_CALL\_STATE\_INFO \*param = (TE\_S\_CALL\_STATE\_INFO \*)Lparam;

if (NULL == m\_incomingCall)

{

return 0L;

}

m\_incomingCall->m\_CallState.Format(\_T("Calling..."));

memcpy(&m\_incomingCall->m\_CallStateInfo, param, sizeof(TE\_S\_CALL\_STATE\_INFO));

::PostMessage(m\_incomingCall->GetSafeHwnd(), WM\_CALL\_UPDATE\_STATE, NULL, NULL);

if (param)

{

delete param;

param = NULL;

}

用户B客户端程序根据是音频来电还是视频来电判断调用TE\_AnswerAudioCall或TE\_AnswerVideoCall接口接听来电。代码如下：

int ret =0;

// TODO:

//音频来电

if (TE\_E\_CALL\_TYPE\_IPAUDIO == m\_CallStateInfo.enCallType)

{

ret = TE\_AnswerAudioCall(m\_CallStateInfo.ulCallID);

}

//视频来电

else if(TE\_E\_CALL\_TYPE\_IPVIDEO == m\_CallStateInfo.enCallType)

{

ret = TE\_AnswerVideoCall(m\_CallStateInfo.ulCallID,1);

}

用户A客户端程序收到TE\_E\_CALL\_EVENT\_ANSWERRSP的回调消息，表示用户B已经接听来电。代码如下：

TE\_S\_CALL\_STATE\_INFO \*param = (TE\_S\_CALL\_STATE\_INFO \*)Lparam;

if (NULL == m\_incomingCall)

{

return 0L;

}

m\_incomingCall->SetTimer(TIME\_ID,SECOND,NULL);

m\_incomingCall->m\_CallState.Format(\_T("Conversation"));

memcpy(&m\_incomingCall->m\_CallStateInfo, param, sizeof(TE\_S\_CALL\_STATE\_INFO));

//设置视频窗口接口

TE\_SetDisplayHandle(param->ulCallID,m\_incomingCall->m\_LocalWindow.GetSafeHwnd(), m\_incomingCall->m\_RemoteWindow.GetSafeHwnd());

::PostMessage(m\_incomingCall->GetSafeHwnd(), WM\_CALL\_UPDATE\_STATE, NULL, NULL);

if (param)

{

delete param;

param = NULL;

}

用户A程序界面提示呼叫接通，开始计算呼叫接通时间。代码如下：

void CTE\_IncomingCall::OnTimer(UINT\_PTR nIDEvent)

{

if(TIME\_ID == nIDEvent)

{

m\_lDuration++;

CString strTimer;

int secod = m\_lDuration%60;

int minte = m\_lDuration/60;

int hour = minte/60;

minte = minte%60;

strTimer.Format("%2d:%2d:%2d",hour,minte,secod);

SetWindowText(strTimer);

}

CDialog::OnTimer(nIDEvent);

}

通话结束时，用户B主动点击挂断，客户端程序调用TE\_DropCall接口结束呼叫。代码如下：

m\_bIsHangup = true;

m\_lDuration = 0;

//结束呼叫接口

TE\_DropCall(m\_CallStateInfo.ulCallID);

m\_lDuration = 0;

KillTimer(TIME\_ID);

m\_CallState.Format(\_T("Call End"));

UpdateData(FALSE);

m\_LocalWindow.ShowWindow(SW\_HIDE);

m\_RemoteWindow.ShowWindow(SW\_HIDE);

m\_isVideo = 0;

OnCancel();

用户A客户端程序接收到呼叫结束通知消息，消息ID为枚举值TE\_E\_CALL\_EVENT\_CLOSE。代码如下：

TE\_S\_CALL\_STATE\_INFO \*param = (TE\_S\_CALL\_STATE\_INFO \*)Lparam;

if (m\_incomingCall != NULL)

{

m\_incomingCall->MessageBox(\_T("Call End"));

m\_incomingCall->KillTimer(TIME\_ID);

m\_incomingCall->m\_lDuration = 0;

m\_incomingCall->m\_CallState.Format(\_T("Call End"));

memcpy(&m\_incomingCall->m\_CallStateInfo, param, sizeof(TE\_S\_CALL\_STATE\_INFO));

::PostMessage(m\_incomingCall->GetSafeHwnd(), WM\_CALL\_UPDATE\_STATE, NULL, NULL);

}

if (param)

{

delete param;

param = NULL;

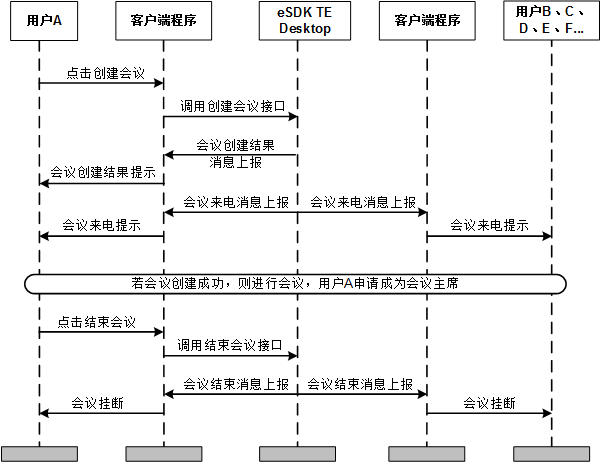
}

----结束

# 场景二：创建会议

业务流程

实现创建会议的时序图如下所示：



实现创建会议的流程说明如下：

用户A点击创建会议，客户端程序调用TE\_CreateConf接口创建会议。

会议来电消息分发给用户A和用户B、C、D、E、F…。

用户A和用户B、C、D、E、F…客户端程序同时接收到会议来电消息，并跳出接听或拒绝窗口，供用户选择加入会议或拒绝入会。

若会议创建成功，则进行会议，用户A申请成为会议主席。结束会议时，用户A选择结束会议，客户端程序调用TE\_EndConf接口结束会议。

会议结束消息分发给用户A和用户B、C、D、E、F…。

用户A和用户B、C、D、E、F…客户端程序同时接收到会议结束消息，挂断会议。

image013

只有会议主席才有权限进行结束会议操作。

实现说明

用户A点击创建会议，客户端程序调用TE\_CreateConf接口创建会议。代码如下：

if (!m\_isLogin)

{

AfxMessageBox(\_T("Please Login in."));

return;

} TE\_S\_CONF\_INVITEDLIST \_contacts;

\_contacts.list[0]="01052414";

\_contacts.list[1]="01052415";

\_contacts.listCount=2;

TE\_CreateConf(&\_contacts);

m\_confInfo.SetWindowText(\_T("等待服务器响应，请稍候..."));

OnCancel();

用户A选择结束会议，客户端程序调用TE\_EndConf接口结束会议。代码如下：

if (TE\_SDK\_Success==TE\_IsConfChairMan())

{

int rest=::MessageBox(NULL,"点击‘是’结束会议，点击‘否’离开会议，点击‘取消’放弃本次操作！","INFO",MB\_YESNOCANCEL);

if(IDYES==rest)

{

TE\_EndConf();

}

else if(IDNO==rest)

{

}

else

{

return;

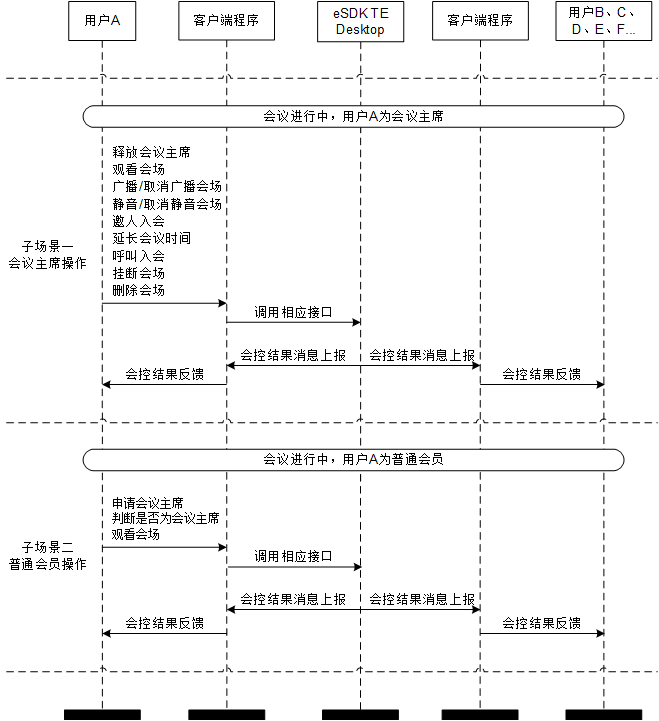
}

}

# 场景三：会控操作

业务流程

实现会控操作的时序图如下所示：



时序图中展示了会控操作的两个子场景，各场景的流程说明分别如下：

子场景一：会议主席进行的会控操作。

用户A若为会议主席，则可以执行释放会议主席、观看会场、广播/取消广播会场、静音/取消静音会场、邀人入会、延长会议时间、呼叫入会、挂断会场、删除会场操作。

执行不同的操作会调用相应的接口，服务器处理操作请求并返回处理结果。

用户A和用户B、C、D、E、F…客户端程序同时接收到会控结果消息，要么弹出结果提示窗口，要么刷新会议列表。

子场景二：普通会员进行的会控操作。

用户A若为普通会员，则可以执行申请会议主席、观看会场操作。

执行不同的操作会调用相应的接口，服务器处理操作请求并返回处理结果。

用户A和用户B、C、D、E、F…客户端程序同时接收到会控结果消息，要么弹出结果提示窗口，要么刷新会议列表。

实现说明

子场景一：会议主席进行的会控操作。

* 释放会议主席，调用TE\_ReleaseConfChairman接口。代码如下：

void CTE\_ConfCtrl::OnMenuReleaseChairman()

{

// TODO: 在此添加命令处理程序代码

if(IDYES==::MessageBox(NULL,"确定释放主席权限？","INFO",MB\_YESNO))

{

TE\_ReleaseConfChairman();

}

}

* 观看会场，调用TE\_WatchSite接口。代码如下：

TE\_WatchSite(&\_confHallList.list[curHall]);

* 广播会场，调用TE\_BroadCastSite接口。代码如下：

TE\_BroadCastSite(&\_confHallList.list[curHall]);

* 取消会场广播，调用TE\_UnBroadCastSite接口。代码如下：

TE\_UnBroadCastSite(&\_confHallList.list[curHall]);

* 静音会场，调用TE\_MuteSite接口。代码如下：

TE\_MuteSite(&\_confHallList.list[curHall]);

* 取消会场静音，调用TE\_UnMuteSite接口。代码如下：

TE\_UnMuteSite(&\_confHallList.list[curHall]);

* 邀人入会，调用TE\_InviteSiteToConf接口。代码如下：

TE\_S\_SITE\_INVITEDLIST \_contacts;

contacts.listCount=0;

int hasSelected=0;

int len = 0;

for(int i = 0 ; i < m\_memLists.GetCount() ; i++)

{

if(m\_memLists.GetCheck(i))

{

CString str;

m\_memLists.GetText(i, str);

len = str.GetLength();

char \* itemstr=new char[len+1];

memset(itemstr,0,len+1);

strncpy(itemstr, str.GetBuffer(len), len);

contacts.list[hasSelected]=itemstr;

contacts.listCount++;

hasSelected++;

}

}

if (hasSelected>0)

{

if(TE\_SDK\_OverMax==TE\_InviteSiteToConf(&\_contacts))

{

GetDlgItem(IDD\_DIALOG\_INVITETOJION)->SetWindowText(\_T("失败：超出会议最大人数"));

}

for(int i=0;i<\_contacts.listCount;i++)

{

delete \_contacts.list[i];

}

}

* 延长会议时间，调用TE\_ExtendConfTime接口。代码如下：

int nIndex = m\_timeList.GetCurSel();

short strTimeValue;

switch(nIndex)

{

case 0:

{

strTimeValue = 30;

}

break;

case 1:

{

strTimeValue = 60;

}

break;

case 2:

{

strTimeValue = 90;

}

break;

case 3:

{

strTimeValue = 120;

}

break;

case 4:

{

strTimeValue = 180;

}

break;

default:

{

strTimeValue = 30;

}

}

TE\_ExtendConfTime(strTimeValue);

* 呼叫入会，调用TE\_CallSiteToConf接口。代码如下：

TE\_CallSiteToConf(&\_confHallList.list[curHall]);

* 挂断会场，调用TE\_HandUpSite接口。代码如下：

TE\_HandUpSite(&\_confHallList.list[curHall]);

* 删除会场，调用TE\_DeleteSite接口。代码如下：

TE\_DeleteSite(&\_confHallList.list[curHall]);

子场景二：普通会员进行的会控操作。

* 申请会议主席，调用TE\_ApplyConfChairman接口。代码如下：

int len = 0;

char passWD[20]={0};

UpdateData(TRUE);

len = m\_edtPassWord.GetLength();

if(len>19)

{

AfxMessageBox("the password too long,please enter again!");

return;

}

strncpy(passWD, m\_edtPassWord.GetBuffer(len), len);

if(TE\_SDK\_Failed==TE\_ApplyConfChairman(passWD))

{

GetDlgItem(IDC\_STATIC3)->SetWindowText(\_T("失败：会议主席已存在"));

}

* 判断是否为会议主席，调用TE\_IsConfChairMan接口。代码如下：

if (TE\_SDK\_Success==TE\_IsConfChairMan())

{

int rest=::MessageBox(NULL,"点击‘是’结束会议；点击‘否’离开会议；点击‘取消’放弃本次操作！","INFO",MB\_YESNOCANCEL);

if(IDYES==rest)

{

TE\_EndConf();

}

else if(IDNO==rest)

{

}

else

{

return;

}

}

* 观看会场，调用TE\_WatchSite接口。代码如下：

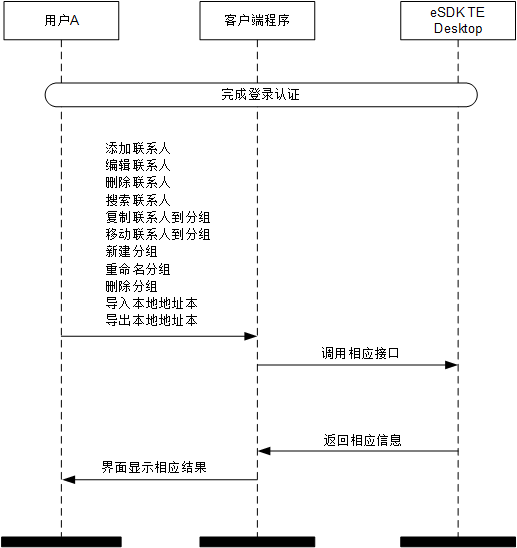
TE\_WatchSite(&\_confHallList.list[curHall]);

# 场景四：通讯录操作

## 通讯录基本操作

业务流程

实现通讯录基本操作的时序图如下所示：



实现通讯录基本操作的流程说明如下：

用户A点击相应控件，客户端程序调用相应接口。

用户A客户端程序根据相应接口返回信息在界面上显示相应结果。

image013

通讯录基本操作包括：添加联系人、编辑联系人、删除联系人、搜索联系人、复制联系人到分组、移动联系人到分组、新建分组、重命名分组、删除分组、导入本地地址本、导出本地地址本。

实现说明

* 添加联系人，调用TE\_AddContact接口。代码如下：

ret = TE\_AddContact(m\_contact);

if(0 == ret)

{

AfxMessageBox("add contact success");

m\_info->OnUpdateRecordList();

}

else if(TE\_SDK\_AddContactFailed == ret)

{

if(MessageBox("contact already exist,if replace","system hint",MB\_ICONEXCLAMATION|MB\_OKCANCEL)!=IDOK)

{

return;

}

* 编辑联系人，调用TE\_EditContact接口。代码如下：

ret = TE\_EditContact(m\_contact);

if(0 == ret)

{

AfxMessageBox("edit contact success");

m\_info->OnUpdateRecordList();

}

else if(TE\_SDK\_GetCustomGroupListFailed == ret)

{

if(MessageBox("contact already exist,if replace","system hint",MB\_ICONEXCLAMATION|MB\_OKCANCEL)!=IDOK)

{

return;

}

* 删除联系人，调用TE\_RemoveContact接口。代码如下：

ret = 0;

//TE\_S\_CONTACT\_INFO contact;

TE\_CHAR m\_contact[512]={0};

strncpy(m\_contact,m\_contactName.c\_str(),65);

// ret = TE\_GetContactByName(m\_contact,contact);

ret = TE\_RemoveContact(m\_contact);

if(TE\_SDK\_Success == ret)

{

AfxMessageBox("delete contact success");

OnUpdateRecordList();

}

else

{

AfxMessageBox("delete contact failed");

}

* 搜索联系人，调用TE\_SearchContact接口。代码如下：

char searchkey[512]={0};

TE\_S\_CONTACTS\_INFO m\_getContact;

m\_getContact.contact = NULL;

m\_getContact.num = 0;

int ret = TE\_SearchContact(searchkey,&m\_getContact);

if（TE\_SDK\_Success== ret）

{

AfxMessageBox((\_T("调用成功 ")));

}

TE\_FreeMemory(m\_getContact.contact);

* 复制联系人到分组，调用TE\_CopyContactToGroup接口。代码如下：

char contactName[65] = {0};

char groupName[65] = {0};

int ret = TE\_CopyContactToGroup(contactName,groupName);

if(0 == ret)

{

AfxMessageBox("copy success");

}

else

{

AfxMessageBox("copy failed");

}

* 移动联系人到分组，调用TE\_MoveContactToGroup接口。代码如下：

char contactName[65] = {0};

char newGroupName[65] = {0};

char oldGroupName[65] = {0};

int ret = TE\_MoveContactToGroup(contactName,newGroupName, oldGroupName);

if(0 == ret)

{

AfxMessageBox("move success");

}

else

{

AfxMessageBox("move failed");

}

* 新建分组，调用TE\_CreateCustomGroup接口。代码如下：

int ret =0;

char GroupName[512]={0};

std::wstring cstrName;

UpdateData(TRUE);

cstrName = m\_demoLg->CstringToWstring(m\_addgroup);

strncpy(GroupName,m\_demoLg->Unicode2Utf8(cstrName).c\_str(),65);

ret = TE\_CreateCustomGroup(GroupName);

if(0 == ret)

{

MessageBox("add group success");

m\_contactdlg->OnUpdateRecordList();

}

else

{

MessageBox(("add group failed"));

}

* 重命名分组，调用TE\_RenameCustomGroup接口。代码如下：

int ret =0,i=0;

char GroupName[512]={0};

UpdateData(TRUE);

char OldGroupName [512] = {0};

int len = m\_RenameGroup.GetLength();

strncpy(GroupName,m\_RenameGroup.GetBuffer(len),len);

strncpy(OldGroupName,m\_reName.c\_str(),65);

ret = TE\_RenameCustomGroup(GroupName,OldGroupName);

if(0 == ret)

{

MessageBox("rename group success");

m\_ContactPoint->OnUpdateRecordList();

}

else

{

MessageBox(("rename group failed"));

}

OnOK();

* 删除分组，调用TE\_RemoveCustomGroup接口。代码如下：

if(0 == ret)

{

for(i=0;i<getGroup.num;i++)

{

cstrName = m\_parent->Utf82Unicode(getGroup.group[i].name);

if(!strcmp(m\_parent->WSToString(cstrName).c\_str() ,m\_contactName.c\_str()))

{

if(MessageBox( "Are you sure delete this group?" ,"system hint" , MB\_ICONINFORMATION|MB\_YESNO)==IDYES)

{

ret = TE\_RemoveCustomGroup(getGroup.group[i].name);

if(0 == ret)

{

AfxMessageBox(\_T("delete this GROUP success"));

OnUpdateRecordList();

}

}

else

{

AfxMessageBox(\_T("delete this group failed"));

}

}

}

}

* 导入本地地址本，调用TE\_ImportFile接口。代码如下：

TE\_S\_GROUPS\_CHAIN\* groupsList = NULL;

TE\_S\_GROUPS\_CHAIN \*groupChainHead= NULL;

TE\_S\_GROUPS\_CHAIN \*groupChainTail= NULL;

int i=0,len = 0;

int rowEnd = m\_ContacList.GetItemCount();

for(i=0;i<rowEnd;i++)

{

TE\_S\_GROUP\_INFO \*tmpgroupInfo = (TE\_S\_GROUP\_INFO \*)malloc(sizeof(TE\_S\_GROUP\_INFO));

memset(tmpgroupInfo, 0, sizeof(TE\_S\_GROUP\_INFO));

TE\_S\_GROUP\_INFO \*groupInfo = NULL;

TE\_S\_CONTACTS\_CHAIN \*contactsChain =(TE\_S\_CONTACTS\_CHAIN\*)malloc(sizeof(TE\_S\_CONTACTS\_CHAIN));

memset(contactsChain,0,sizeof(TE\_S\_CONTACTS\_CHAIN));

if(groupInfo = findGroupInChain(tmpgroupInfo->groupName,groupChainHead))

{

InsertContactInGroup(groupInfo, contactsChain);

if(tmpgroupInfo)

{

free(tmpgroupInfo);

tmpgroupInfo = NULL;

}

}

else

{

groupChainTail = AddGroupChain(groupChainHead,tmpgroupInfo);

if(!groupChainHead)

{

groupChainHead = groupChainTail;

}

InsertContactInGroup(groupChainTail->group, contactsChain);

}

}

groupsList = groupChainHead;

int ret = TE\_ImportFile( groupsList);

if(0 == ret)

{

AfxMessageBox(\_T("Import file success"));

}

else

{

AfxMessageBox(\_T("Import file filed"));

}

while(groupsList)

{

while(groupsList->group->contactsChain)

{

TE\_S\_CONTACTS\_CHAIN\* tempContactsChain = groupsList->group->contactsChain;

groupsList->group->contactsChain = groupsList->group->contactsChain->next;

TE\_FreeMemory(tempContactsChain);

}

TE\_S\_GROUPS\_CHAIN\* tempGroupsChain = groupsList;

groupsList = groupsList->next;

TE\_FreeMemory(tempGroupsChain);

}

* 导出本地地址本，调用TE\_ExportExcelFile接口。代码如下：

char FileName[512]={0};

int leng = FilePathName.GetLength();

strncpy(FileName,FilePathName.GetBuffer(leng),leng);

// 取文件的后缀名

CString strPostFix = FilePathName.Right(POSTFIX\_COUNT);

if (0 == strPostFix.Compare(\_T("lsx")))

{

strPostFix = FilePathName.Right(POSTFIX\_COUNT + 1);

}

//不是Excel文件就是CSV文件

if ((EXCEL\_POSTFIX == strPostFix) || (EXCEL2007\_POSTFIX == strPostFix))

{

ret = TE\_ExportExcelFile(FileName,&groupsList,1);

if(0 == ret)

{

AfxMessageBox("Export Excel File Success");

}

else

{

AfxMessageBox("Export Excel File Failed");

}

}

else if (CSV\_POSTFIX == strPostFix)

{

ret = TE\_ExportExcelFile(FileName,&groupsList,2);

if(0 == ret)

{

AfxMessageBox("Export Csv File Success");

}

else

{

AfxMessageBox("Export Csv File Failed");

}

}

else

{

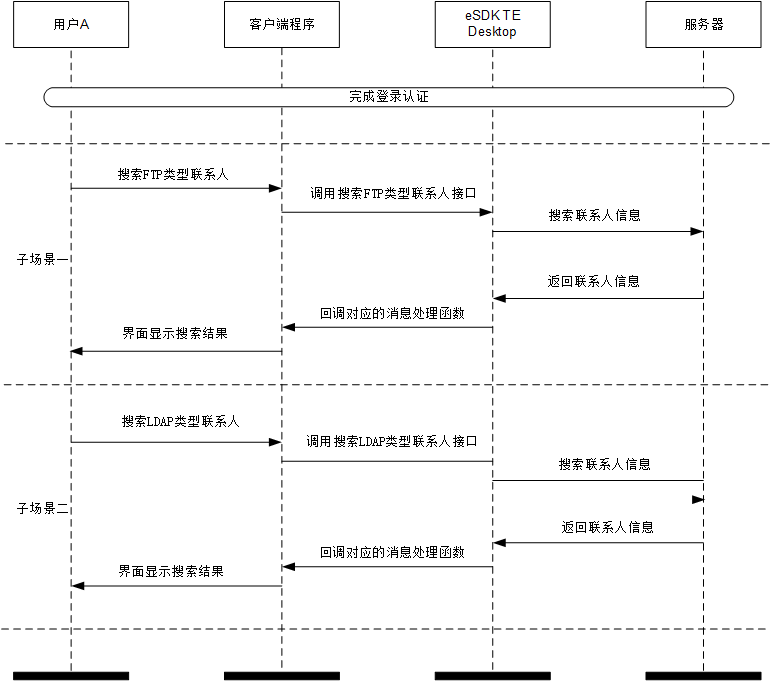
}

OnOK();

## 搜索企业通讯录联系人

业务流程

实现搜索企业通讯录联系人的时序图如下所示：



实现搜索企业通讯录联系人的时序图中包括两个子场景，各子场景的流程说明如下：

子场景一：搜索FTP类型联系人。

用户A点击搜索FTP类型联系人控件，客户端程序调用搜索FTP类型联系人接口。

用户A客户端程序根据接口返回信息在界面上显示相应结果。

子场景二：搜索LDAP类型联系人。

用户A点击搜索LDAP类型联系人控件，客户端程序调用搜索LDAP类型联系人接口。

用户A客户端程序根据接口返回信息在界面上显示相应结果。

实现说明

子场景一：搜索FTP类型联系人。

搜索FTP类型联系人，调用TE\_SearchFtp接口。代码如下：

char keyword[512]={""};

TE\_S\_CONTACTS\_INFO contactsList;

contactsList.contact = NULL;

contactsList.num = 0;

int ret = TE\_SearchFtp(keyword,&contactsList);

if(TE\_SDK\_Success== ret)

{

AfxMessageBox((\_T("调用成功")));

}

TE\_FreeMemory(contactsList.contact);

子场景二：搜索LDAP类型联系人。

搜索LDAP类型联系人，调用TE\_SearchLdap接口。代码如下：

char keyword[512]={""};

TE\_SearchLdap( keyword); //调用接口

//进入回调

TE\_S\_ENTERPRISE\_RESULT\* wParam = (TE\_S\_ENTERPRISE\_RESULT \*)Lparam;

if(!wParam)

{

return -1;

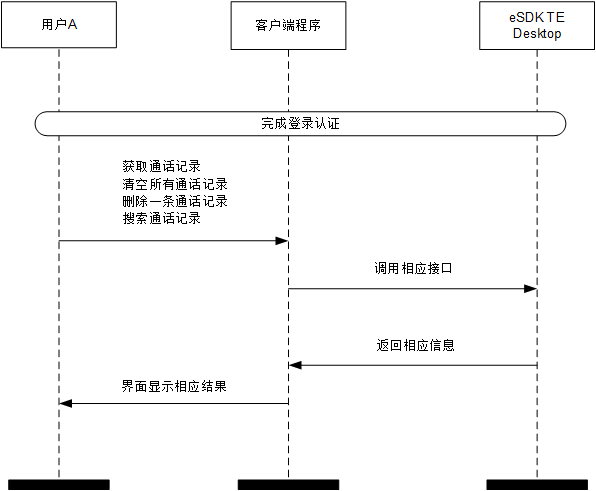
}

OnShowEnterpriseResult(&wParam->enterpriseContacts);

## 历史通话记录操作

业务流程

实现历史通话记录操作的时序图如下所示：



实现历史通话记录操作的流程说明如下：

用户A点击相应控件，客户端程序调用相应的接口。

用户A客户端程序根据相应接口返回信息在界面上显示相应结果。

实现说明

* 获取通话记录，调用TE\_GetRecentCallRecords接口。代码如下：

TE\_S\_PERSONS\_INFO historyInfo;

historyInfo.historyRecords = NULL;

historyInfo.num = -1;

int ret = TE\_GetRecentCallRecords(&historyInfo);

if（TE\_SDK\_Success== ret）

{

AfxMessageBox((\_T("调用成功 ")));

}

* 清空所有通话记录，调用TE\_ClearAllCallRecord接口。代码如下：

int ret = TE\_ClearAllCallRecord();

if（TE\_SDK\_Success== ret）

{

AfxMessageBox((\_T("调用成功 ")));

}

* 删除一条通话记录，调用TE\_DeleteOneCallRecord接口。代码如下：

char histroyRecordId[512] = {"1"};

int ret = TE\_DeleteOneCallRecord(histroyRecordId);

if（TE\_SDK\_Success== ret）

{

AfxMessageBox((\_T("调用成功 ")));

}

* 搜索通话记录，调用TE\_SearchHistoryRecords接口。代码如下：

char searchKey[512]={0};

TE\_S\_PERSONS\_INFO historyRecords;

historyRecords.historyRecords = NULL;

historyRecords.num = 0;

int ret = TE\_SearchHistoryRecords(searchKey,&historyRecords);

if（TE\_SDK\_Success== ret）

{

AfxMessageBox((\_T("调用成功 ")));

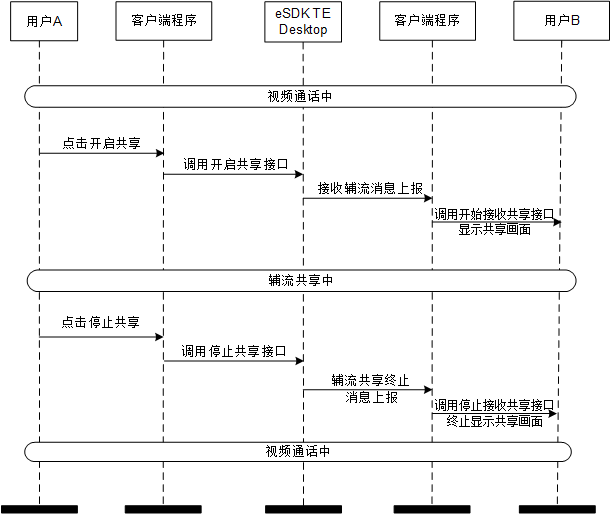
}

TE\_FreeMemory(historyRecords.historyRecords);

# 场景五：辅流控制

业务流程

实现辅流控制的时序图如下所示：



实现辅流控制的流程说明如下：

用户A和用户B正在进行视频通话。用户A点击桌面共享或程序共享，客户端程序调用TE\_StartLecture接口开启共享。

辅流变化结果通过回调返回给用户A和用户B。

用户B客户端程序接收辅流变化结果，判断并回调接收共享函数，在函数中调用TE\_StartRecvLecture接口开始接收共享。

用户B客户端程序界面显示对方的共享画面。

用户A点击停止共享，客户端程序调用TE\_StopLecture接口停止共享。

辅流变化结果通过回调返回给用户A和用户B。

用户B客户端程序接收辅流变化结果，判断并回调停止共享函数，在函数中调用TE\_StopRecvLecture接口停止接收共享。

用户B客户端程序界面终止显示共享画面。

image013

用户B同样可以执行开启/停止共享操作，此处仅以用户A为例进行说明。

实现说明

用户A调用TE\_StartLecture接口开启共享。代码如下：

m\_dataShareWind= ::GetDesktopWindow();//被共享的桌面或程序句柄

//设置被共享的桌面或程序句柄，必须在调用TE\_StartLecture接口之前调用

TE\_SetShareWindHandle(m\_CallStateInfo.ulCallID,m\_dataShareWind);

TE\_StartLecture(m\_CallStateInfo.ulCallID);//开启共享

辅流变化结果通过回调返回给用户A和用户B。辅流协商结果上报事件ID为枚举值TE\_E\_EVT\_BFCPMODIFIED\_RESULT。代码如下：

//回调函数中，判断若是辅流协商结果上报事件ID: TE\_E\_EVT\_BFCPMODIFIED\_RESULT，则处理ID为WM\_IM\_BFCP\_MODIFIED\_RESULT的消息。

case TE\_E\_EVT\_BFCPMODIFIED\_RESULT:

{

TE\_S\_CALL\_STATE\_INFO \*pCallStateInfo = new TE\_S\_CALL\_STATE\_INFO;

CTE\_DemoDlg \*pWnd =NULL;

pWnd = (CTE\_DemoDlg \*)pUserData;

memcpy(pCallStateInfo, pEventBuf, sizeof(TE\_S\_CALL\_STATE\_INFO));

::PostMessage(pWnd->GetSafeHwnd(),WM\_IM\_BFCP\_MODIFIED\_RESULT,

NULL, (LPARAM)pCallStateInfo);

break;

}

//WM\_IM\_BFCP\_MODIFIED\_RESULT为消息对应的处理函数

LRESULT CTE\_DemoDlg::OnBFCPModifiedResult(WPARAM /\*wParam\*/, LPARAM lParam)

{

TE\_S\_CALL\_STATE\_INFO \*param = (TE\_S\_CALL\_STATE\_INFO \*)lParam;

//调用辅流变化结果处理接口TE\_BFCPModifiedResult

TE\_BFCPModifiedResult(param->ulCallID,param->sessionType);

if (param)

{

delete param;

param = NULL;

}

return 0L;

}

用户B客户端程序接收到接收共享回调消息，消息ID为枚举值TE\_E\_DATA\_EVENT\_DATARECIEVE。代码如下：

TE\_S\_CALL\_STATE\_INFO \*param = (TE\_S\_CALL\_STATE\_INFO \*)Lparam;

if (NULL == m\_incomingCall)

{

return 0L;

}

memcpy(&m\_incomingCall->m\_CallStateInfo, param, sizeof(TE\_S\_CALL\_STATE\_INFO));

//调用设置显示共享窗口的句柄接口

TE\_SetDisplayShareHandle(param->ulCallID,m\_incomingCall->m\_displayShareWind.GetSafeHwnd());

::PostMessage(m\_incomingCall->GetSafeHwnd(), WM\_CALL\_UPDATE\_STATE, NULL, NULL);

if (param)

{

delete param;

param = NULL;

}

//收到上面代码处理的消息后，接收消息的窗口在对应的消息处理中调用接收共享接口

//调用接收共享接口

TE\_StartRecvLecture(m\_CallStateInfo.ulCallID);

用户A客户端程序调用TE\_StopLecture接口停止共享。代码如下：

//停止共享

TE\_StopLecture(m\_CallStateInfo.ulCallID);

用户B客户端程序接收到停止共享回调消息，消息ID为枚举值TE\_E\_DATA\_EVENT\_DATASTOPED。代码如下：

TE\_S\_CALL\_STATE\_INFO \*param = (TE\_S\_CALL\_STATE\_INFO \*)Lparam;

if (NULL == m\_incomingCall)

{

if (param)

{

delete param;

param = NULL;

}

return 0L;

}

memcpy(&m\_incomingCall->m\_CallStateInfo, param, sizeof(TE\_S\_CALL\_STATE\_INFO));

::PostMessage(m\_incomingCall->GetSafeHwnd(), WM\_CALL\_UPDATE\_STATE, NULL, NULL);

if (param)

{

delete param;

param = NULL;

}

//收到上面代码处理的消息后，接收消息的窗口在对应的消息处理中调用停止接收共享接口

//调用停止接收共享接口

TE\_StopRecvLecture(m\_CallStateInfo.ulCallID);

----结束