# 恩布企业 IM 云通讯平台

## PC SDK 开发手册

| 版本日期 | 作者 | 内容 |
| --- | --- | --- |
| **2012-08-18** | HD | 文档初稿。 |
| **2014-09-18** | HD | 重新整理。 |
| **2014-09-30** | HD | 增加 EB_LoadGroup，EB_JoinGroup 接口，EB_VideoAck 增加 nToUserId 参数，帮助实现视频会议功能。 |
| **2014-11-18** | HD | 增加 EB_CanEditMyBaseMemberInfo 接口。 |
| **2014-11-25** | HD | 增加 EB_IsEnterpriseUser 判断是否企业用户功能接口。 |
| **2014-11-29** | HD | 1.增加 EB_Save2CloudDrive 把离线文件存入我的云盘功能接口。<br>2.增加 EB_GetSubscribeFuncInfo 接口。 |
| **2015-01-21** | HD | 增加联系人分组管理接口和申请添加好友接口； |
| **2015-04-11** | HD | 增加 EB_GetSubscribeFuncSize 接口； |

恩布互联、恩布云通信平台、entboost 等商标归
深圳恩布网络科技有限公司，版本所有
2013 - 2015

# 1.概述

## 1.1. 功能描述

| 功能 | 描述 |
| --- | --- |
| **富文本格式** | 支持文本+图片混合排版传输 |
| **图片** | 支持 BMP、JPG、PNG、ICO、GIF 等所有格式图片 |
| **表情** | 支持企业（部门）自定义动画表情 |
| **文件** | 支持文件秒传功能，离线文件，群组文件传输等 |
| **视频（语音）通话** | 支持一对一视频（或语音）通话 |
| **多方视频（语音）会议** | 支持多方（群组、部门）视频会议和语音会议，不限制人数 |
| **企业组织结构** | 支持多级企业组织结构，部门、项目组、群组和临时讨论组等 |
| **通讯录** | 支持可分组通讯录（联系人）资料管理 |
| **群组通讯** | 支持一对多，多对多的群组（部门）通讯 |
| **离线功能** | 支持离线收发消息功能 |
| **云资源共享** | 支持个人云盘、群（部门）云盘；在线文件等资源共享 |
| **电子名片** | 支持外部企业，电子名片显示功能 |
| **游客功能** | 支持免注册登录，游客使用功能 |
| **屏幕截图** | 支持屏蔽截图功能 |
| **全网互联通讯** | 加入公有云服务，支持全网互联通讯 |
| **可扩展通信接口** | 支持自定义、可扩展通信接口，满足企业定制业务需求 |

利用恩布互联 SDK，可以快速开发完整的企业版本即时通讯软件，集成云通讯功能到企业内部系统中；

## 1.2. 恩布互联SDK能开发什么应用

A、能够开发完整的企业版本类 QQ、类微信软件、在线客服等产品；

B、能够开发各种通信组件，集成到企业现有软件、管理系统中；

C、开发 PC 端、WEB 浏览器端和 IOS、ANDROID 手机端的融合通信产品；

## 1.3. 开发流程

A、下载全套客户端源码，编译、学习和试用；

B、免费申请开发者 APP ID 和 APP KEY；

C、编码、开发、测试；

D、 申请加入公有云、或私有云部署，完成开发；

## 1.4. 支持开发环境

- 标准 C++接口，.h、.lib 和.dll 开发环境

  支持 VC++2008（9.0 版本）直接使用；

- 标准 COM 组件接口

  支持所有其他 C++开发工具（如 VC6、VC2003、VC2010、C++Builder 等）；

  支持 VB、Delphi 等开发语言、开发工具；

支持 JavaScript、VBScript、ASP 等脚本语言开发环境；

## 1.5. SDK文件列表

| 文件 | 描述 |
| --- | --- |
| **ebcore.dll** | *云通信主业务 COM 组件 |
| **ebvideoroom.dll** | *视频通讯接口函数 COM 组件 |
| **EBAppClient.h** | *云通信主业务接口函数头文件 |
| **EBCallbackInterface.h** | *云通信主业务回调函数头文件 |
| **videoroom.h** | 视频通讯接口函数头文件 |
| **videoroomdefine.h** | 视频通讯类型定义头文件 |
| **include\chatroomobject.h** | 聊天内容信息类定义头文件 |
| **include\chatroomhandle.h** | 聊天消息回调函数头文件 |
| **include\eb_define.h** | 云通信基础类型定义头文件 |
| **include\eb_object.h** | 云通信基础对象类定义头文件 |
| **include\ebstring.h** | CEBString 字符串封装类头文件 |
| **libebum.lib** | 云通信核心基础库 LIB 文件 |
| **libebum.dll** | 云通信核心基础库文件 |
| **libebcm.dll** | 云通信核心基础库文件 |
| **videoroom.lib** | 云通信视频核心基础库 LIB 文件 |
| **videoroom.dll** | 云通信视频核心基础库文件 |
| **swscale-2.dll** | 视频基础库文件 |

| | |
|---|---|
| **avcodec-53.dll** | 视频基础库文件 |
| **avformat-53.dll** | 视频基础库文件 |
| **avutil-51.dll** | 视频基础库文件 |

# 2.接口函数说明

## 2.1. 接口描述

### 2.1.1. 使用标准C++开发接口

VC++2008（9.0），支持使用 C++开发接口；

包括头文件：

```
#include "videoroom.h"
#include "EBAppClient.h"
```

链接 LIB 文件：

```
#pragma comment(lib, "videoroom.lib")
#pragma comment(lib, "libebum.lib")
```

定义实例即可使用：

```
CEBAppClient theEBAppClient;
theEBAppClient.EB_Init(sAddress.c_str());
theEBAppClient.EB_SetMsgHwnd(this->GetSafeHwnd());
theEBAppClient.EB_SetDevAppId("[your app id]","[your app key]",true);
...
```

回调文件名：EBCallbackInterface.h

所有云通信平台操作都为异步机制，通过 EBCallbackInterface.h 定义的类和消息返回消息事件。

返回 int 函数（如 int func(...)），返回 0 表示操作成功，但不一定表示业务成功，业务结果看异步回设结果；

### 2.1.2. 使用COM开发接口

VC++导入 COM 组件；

```
#import "..\build\ebvideoroom.dll" no_namespace
#import "..\build\ ebcore.dll" no_namespace
```

定义接口：

```
CComPtr<IEBClientCore>      theEBClientCore;
```

初始化 COM 环境：

```
::CoInitialize(NULL);
```

创建 COM 实例，开发使用 COM 接口：

```
CoCreateInstance(__uuidof(EBClientCore),NULL,CLSCTX_INPROC_SERVER,__uuidof(IEBClientCore)
,(void**)&theEBClientCore);
theEBClientCore->EB_Init(sAddress.c_str(),"");
this->DispEventAdvise(theEBClientCore); // 连接点
theEBClientCore->EB_SetDevAppId("[your app id]"," [your app key]",VARIANT_TRUE);
…
```

所有 COM 接口，返回标准 HRESUL 值；

其他开发工具请参照使用；

### 2.1.3. 脚本语言使用COM接口例子

```
<object                    classid="clsid:B26BB29E-A61D-48E7-8068-6CF59FDD7680"
id=objEBCore>
</object>
<script type="text/javascript">
    objEBCore.EB_Init("[ip-server]","")
    …
    objEBCore.EB_UnInit()
```

```
</script>
```

或者

```
<script type="text/vbscript">
    Dim objEBCore
    Set objEBCore = CreateObject("ebcore.EBClientCore.1")
    objEBCore.EB_Init "[ip-server]",""
    …
    objEBCore.EB_UnInit
</script>
```

### 2.1.4. 更多开发工具应用及使用例子请访问www.entboost.com

## 2.2. 应用环境函数

### 2.2.1. EB_Init

```
void EB_Init(const char * sAddress,const char* sInitParameter="");
COM 接口：EB_Init([in] BSTR sAddress, [in] BSTR sInitParameter);
```

初始化 SDK 应用环境，SDK 必须先初始化应用环境，才能正常使用；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sAddress** | | 通信平台服务器地址<br>格式:ip:port 如 192.168.10.18:18012 |
| **const char* sInitParameter** | 可选 | 初始化参数 |

### 2.2.2. EB_IsInited

bool EB_IsInited(void) const;

COM 接口：EB_IsInited([out, retval] VARIANT_BOOL* pVal);

判断是否已经初始化应用环境；

### 2.2.3. EB_SetCallBack

void EB_SetCallBack(CEBCallbackInterface * pCallBack);

COM 接口：该接口使用 COM 事件连接点；

设置事件回设监听类地址；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **CEBCallbackInterface * pCallBack** | | 上层应用事件监听类；<br>NULL，空为注销 |

### 2.2.4. EB_SetMsgHwnd

void EB_SetMsgHwnd(HWND pHwnd);

COM 接口，该接口使用 COM 事件连接点；

设置事件回设监听窗口句柄；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **HWND pHwnd** | | 上层应用事件监听窗口句柄；<br>NULL，空为注销 |

### 2.2.5. EB_UnInit

void EB_UnInit(void);

COM 接口：EB_UnInit(void);

注释 SDK 应用环境；

### 2.2.6. EB_GetLastStateCode

EB_STATE_CODE EB_GetLastStateCode(void) const;

COM 接口参数：EB_LastStateCode([out, retval] LONG* pVal);

获取最后操作状态码；( 状态码值及表示，见附录。)

### 2.2.7. EB_GetAddress

std::string EB_GetAddress(void) const;

COM 接口参数：EB_Address([out, retval] BSTR* pVal);

返回通信平台服务器地址；

### 2.2.8. EB_GetAppPath

std::string EB_GetAppPath(void) const;

COM 接口参数：EB_AppPath([out, retval] BSTR* pVal);

返回应用程序 APP 路径；

### 2.2.9. EB_GetResourcePath

std::string EB_GetResourcePath(void) const;

COM 接口参数：EB_ResourcePath([out, retval] BSTR* pVal);

返回资源文件存放路径；

### 2.2.10. EB_GetAppDatePath

std::string EB_GetAppDatePath(void) const;

COM 接口参数：EB_AppDataPath([out, retval] BSTR* pVal);

返回应用程序 App 数据路径；

### 2.2.11. EB_GetAppDateTempPath

std::string EB_GetAppDateTempPath(void) const;

COM 接口参数：EB_AppDataTempPath([out, retval] BSTR* pVal);

返回应用程序 App 临时数据路径；

### 2.2.12. EB_CheckVersion

int EB_CheckVersion(const char* sClientVersion);

COM 接口：EB_CheckVersion([in] BSTR sClientVersion);

检查最新版本 ;如果有新版本 ,自动下载更新包文件 ,下载成功会收到 onNewVersion() 或 EB_WM_NEW_VERSION 通知；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sClientVersion** | | 当前版本 |

## 2.3. 开发者ID函数

### 2.3.1. 描述

一个 APP ID 代表一个应用 ;一个开发者 ,根据商业许可可以申请一个或多个 APP ID；

SDK 必须设置正确的开发者 APP ID 和 APP KEY ,通过验证成功 ,获取应用在线 KEY，才能正常的注册和登录等；

### 2.3.2. EB_SetDevAppId

int EB_SetDevAppId(const char* sAppId,const char* sAppKey,bool bReLoadAppOnlineKey=true);

COM 接口：EB_SetDevAppId([in] BSTR sAppId, [in] BSTR sAppKey, [in] VARIANT_BOOL bReLoadAppOnlineKey);

设置开发者 APP ID 和 APP KEY；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sAppId** | | 开发者 APP ID |
| **const char* sAppKey** | | 开发者 APP KEY |
| **bool bReLoadAppOnlineKey** | true | 是否访问服务器，验证 APP ID 和 APP KEY；true/false |

### 2.3.3. EB_SetDevAppOnlineKey

void EB_SetDevAppOnlineKey(const char* sAppId, const char* sAppOnlineKey);

COM 接口：EB_SetDevAppOnlineKey([in] BSTR sAppId, [in] BSTR sAppOnlineKey);

设置开发者 APP ID 和已经验证成功的应用在线 KEY；（这个用于在其他地方验证）

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sAppId** | | 开发者 APP ID |
| **const char* sAppOnlineKey** | | 验证成功应用在线 KEY |

### 2.3.4. EB_GetDevAppId

std::string EB_GetDevAppId(void) const;

COM 接口：EB_GetDevAppId([out,retval] BSTR* pVal);

取出开发者 APP ID；

### 2.3.5. EB_GetDevAppOnlineKey

std::string EB_GetDevAppOnlineKey(void) const;

COM 接口：EB_GetDevAppOnlineKey([out,retval] BSTR* pVal);

取出已经验证成功的应用在线 KEY；

# 2.4. AP应用中心函数

## 2.4.1. 描述

利用 AP 应用中心函数，可以实现各种自定义消息的传递；

## 2.4.2. EB_LogonAppCenter

int EB_LogonAppCenter(void);

COM 接口：EB_LogonAppCenter(void);

申请以当前开发者 ID 登录到应用中心；

登录成功，如果有离线消息，会收到 onAPMsgInfo 事件或 EB_WM_AP_MSG 消息；

保持登录状态，同样可以收到其他应用发送的消息；

## 2.4.3. EB_LogoutAppCenter

int EB_LogoutAppCenter(void);

COM 接口：EB_LogoutAppCenter(void);

退出应用中心；

## 2.4.4. EB_SendAPMsg

int EB_SendAPMsg(const EB_APMsgInfo& pAPMsgInfo, bool bSaveOffMsg);

COM 接口：EB_SendAPMsg([in] IDispatch* pAPMsgInfoInterface, [in] VARIANT_BOOL bSaveOffMsg);

发送消息给指定 APP 应用；

| 参数 | | 默认值（非空） | 描述 |
|---|---|---|---|
| **const pAPMsgInfo** | **EB_APMsgInfo&** | | 应用消息结构 |

| | |
|---|---|
| **bool bSaveOffMsg** | 应用不在线时，是否保存离线消息<br>true/false |

### 2.4.5. EB_APMsgAck

void EB_APMsgAck(eb::bigint nMsgId);

COM 接口：EB_APMsgAck([in] ULONGLONG nMsgId);

响应收到 APP 应用消息，收到 APP 消息，必须进行响应；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nMsgId** | | 消息 ID |

## 2.5. 注册验证函数

### 2.5.1. EB_Register

int EB_Register(const char* sAccount, const char* sPassword, const char* sUserName="", const char* sEnterpriseName="");

COM 接口：EB_Register([in] BSTR sAccount, [in] BSTR sPassword, [in] BSTR sUserName, [in] BSTR sEnterpriseName);

注册一个新帐号或企业管理员帐号；

注册成员会收到一条邮箱地址验证邮件，验证成功可以正常登录使用，验证邮件 24 小时内有效，超过 24 小时，需要重新注册。

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **const char* sAccount** | | 帐号邮箱 |
| **const char* sPassword** | | 帐号密码 |
| **const char* sUserName** | “” | 用户名称<br>不填写跟帐号邮件一样 |
| **const char* sEnterpriseName** | “” | 公司名称<br>填写：注册企业管理员帐号<br>不填写：注册个人帐号 |

### 2.5.2. EB_LogonByVisitor

int EB_LogonByVisitor(void);

COM 接口：EB_LogonByVisitor(void);

游客登录；

### 2.5.3. EB_LogonByAccount

int EB_LogonByAccount(const char * sAccount, const char * sPassword, EB_USER_LINE_STATE nNewLineState=EB_LINE_STATE_ONLINE);

COM 接口：EB_LogonByAccount([in] BSTR sAccount, [in] BSTR sPassword, [in] LONG nNewLineState);

用户登录；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sAccount** | | 帐号邮箱 |
| **const char* sPassword** | | 帐号密码 |
| **EB_USER_LINE_STATE nNewLineState** | EB_LINE_STATE_ONLINE | 在线状态 |

### 2.5.4. EB_LogonOAuth

int EB_LogonOAuth(const char * sAccount="",EB_USER_LINE_STATE nNewLineState=EB_LINE_STATE_ONLINE);

COM 接口：EB_LogonOAuth([in] BSTR sAccount, [in] LONG nNewLineState);

请求开放验证登录；请求成功，会返回一串开放验证 URL 链接，使用浏览器打开链接，输入帐号、密码，登录验证；登录成功客户端会收到登录成功事件；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sAccount** | “” | 帐号邮箱 |
| **EB_USER_LINE_STATE nNewLineState** | EB_LINE_STATE_ONLINE | 在线状态 |

### 2.5.5. EB_ReLogon

int EB_ReLogon(void);

COM 接口：EB_ReLogon(void);

重新登录，主要用于用户电脑休眠、挂起，或网络断开重启时，直接重新登录；

### 2.5.6. EB_IsLogoned

bool EB_IsLogoned(void) const;

COM 接口参数：EB_IsLogoned([out, retval] VARIANT_BOOL* pVal);

返回是否已经登录成功；

### 2.5.7. EB_IsLogonVisitor

bool EB_IsLogonVisitor(void) const;

COM 接口参数：EB_IsLogonVisitor([out, retval] VARIANT_BOOL* pVal);

返回是否游客登录；

### 2.5.8. EB_Logout

void EB_Logout(void);

COM 接口：EB_Logout(void);

用户退出；

## 2.6. 基本信息管理函数

### 2.6.1. EB_LoadInfo

void EB_LoadInfo(void);

COM 接口：EB_LoadInfo(void);

登录成功后，加载离线信息等；

登录成功后，先调用 EB_LoadOrg()加载企业组织结构、个人群组等数据，调用 EB_LoadContact()加载通讯录，后再调用 EB_LoadInfo()，

### 2.6.2. EB_GetMyAccountInfo

bool EB_GetMyAccountInfo(EB_AccountInfo* pOutAccountInfo) const;

COM 接口：EB_GetMyAccountInfo([out,retval] IEB_AccountInfo** pVal);

返回登录帐号信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **EB_AccountInfo* pOutAccountInfo** | | [输出]登录帐号信息 |

### 2.6.3. EB_GetMyDefaultMemberCode

bool EB_GetMyDefaultMemberCode(eb::bigint& pOutDefaultMemberCode) const;

COM 接口参数：EB_MyDefaultMemberCode([out, retval] ULONGLONG* pVal);

获取默认群组（部门）成员编号

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint& pOutDefaultMemberCode** | | [输出]默认群组（部门）成员编号 |

### 2.6.4. EB_SetMyDefaultMemberCode

bool EB_SetMyDefaultMemberCode(eb::bigint nNewDefaultMemberCode);

COM 接口参数：EB_MyDefaultMemberCode([in] ULONGLONG newVal);

设置默认群组（部门）成员编号；

默认群组（部门）成员编号，用于在外部呼叫中自动显示电子名片；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |

| | |
|---|---|
| **eb::bigint sNewDefaultMemberCode** | 默认群组（部门）成员编号 |

## 2.6.5. EB_GetMyDefaultMemberInfo

bool EB_GetMyDefaultMemberInfo(EB_MemberInfo* pOutmemberInfo) const;

COM 接口：EB_GetMyDefaultMemberInfo([out,retval] IEB_MemberInfo** pVal);

返回默认群组（部门）成员信息；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **EB_MemberInfo* pOutMemberInfo** | | [输出]默认群组（部门）成员信息 |

## 2.6.6. EB_GetMyDefaultMemberHeadFile

std::string EB_GetMyDefaultMemberHeadFile(void) const;

COM 接口：EB_GetMyDefaultMemberHeadFile([out,retval] BSTR* pVal);

返回默认群组（部门）成员头像文件；

## 2.6.7. EB_GetMyAccountSetting

int EB_GetMyAccountSetting(void) const;

COM 接口参数：EB_MyAccountSetting([out, retval] LONG* pVal);

返回个人设置；个人设置值，见 EB_SETTING_VALUE 定义；

## 2.6.8. EB_SetMyAccountSetting

bool EB_SetMyAccountSetting(int nNewSetting);

COM 接口参数：EB_MyAccountSetting([in] LONG newVal);

更改个人设置；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **int nNewSetting** | | 个人设置 |

### 2.6.9. EB_GetLogonAccount

std::string EB_GetLogonedAccount(void) const;

COM 接口参数：EB_LogonAccount([out, retval] BSTR* pVal);

返回登录帐号；

### 2.6.10. EB_GetLogonUserId

eb::bigint EB_GetLogonedUserId(void) const;

COM 接口参数：EB_LogonUserId([out, retval] ULONGLONG* pVal);

返回登录用户编号；

### 2.6.11. EB_GetDescription

std::string EB_GetDescription(void) const;

COM 接口参数：EB_Description([out, retval] BSTR* pVal);

返回登录帐号备注信息；

### 2.6.12. EB_SetDescription

bool EB_SetDescription(const char* sNewDescription);

COM 接口参数：EB_Description([in] BSTR newVal);

更改帐号备注信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sNewDescription** | | 备注信息 |

### 2.6.13. EB_GetLineState

EB_USER_LINE_STATE EB_GetLineState(void)const;

COM 接口参数：EB_LineState([out, retval] LONG* pVal);

返回我的在线状态；

## 2.6.14. EB_SetLineState

bool EB_SetLineState(EB_USER_LINE_STATE nNewLineState);

COM 接口参数：EB_LineState([in] LONG newVal);

更改我的在线状态；所有在线群组成员（包括自己）会收到 onUserStateChange()或 EB_WM_USER_STATE_CHANGE 通知；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **EB_USER_LINE_STATE nNewLineState** | | 在线状态 |

## 2.6.15. EB_GetUserName

std::string EB_GetUserName(void) const;

COM 接口参数：EB_UserName([out, retval] BSTR* pVal);

返回帐号用户名称；

## 2.6.16. EB_SetUserName

bool EB_SetUserName(const char* sNewUserName);

COM 接口参数：EB_UserName([in] BSTR newVal);

修改帐号用户名称；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sNewUserName** | | 用户名称 |

## 2.6.17. EB_GetPassword

std::string EB_GetPassword(void) const;

COM 接口参数：EB_Password([out, retval] BSTR* pVal);

返回帐号密码；

### 2.6.18. EB_SetPassword

bool EB_SetPassword(const char* sNewPassword, const char* sOldPassword);

COM 接口参数：EB_Password([in] BSTR newVal, [in] BSTR oldVal);

修改密码；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sNewPassword** | | 新密码 |
| **const char* sOldPassword** | | 旧密码 |

## 2.7. 权限管理函数

### 2.7.1. EB_CanEditEnterprise

bool EB_CanEditEnterprise(eb::bigint nEnterpriseCode=0) const;

COM 接口参数：EB_CanEditEnterprise([in] ULONGLONG NEnterpriseCode, [out, retval] VARIANT_BOOL* pVal);

判断是否有编辑企业资料权限；只有企业管理员才有权限编辑；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nEnterpriseCode** | 0 | 企业代码，0 为判断当前默认企业 |

### 2.7.2. EB_CanEditGroupInfo

bool EB_CanEditGroupInfo(eb::bigint nEnterpriseCode,eb::bigint nGroupCode) const;

COM 接口参数：EB_CanEditGroupInfo([in] ULONGLONG nEnterpriseCode, [in] ULONGLONG nGroupCode, [out, retval] VARIANT_BOOL* pVal);

判断是否有编辑群组（部门）资料权限；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nEnterpriseCode** | | 企业代码，0 为判断当前默认企业 |
| **eb::bigint nGroupCode** | | 群组（部门）编号，0 为判断是否有新建 |

| | |
|---|---|
| | 群组（部门）权限 |

## 2.7.3. EB_CanDeleteGroupInfo

bool EB_CanDeleteGroupInfo(eb::bigint nGroupCode) const;

COM 接口参数：EB_CanDeleteGroupInfo([in] ULONGLONG nGroupCode, [out, retval] VARIANT_BOOL* pVal);

判断是否有删除群组（部门）资料权限；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

## 2.7.4. EB_CanEditMemberInfo

bool EB_CanEditMemberInfo(eb::bigint nGroupCode,eb::bigint nMemberUserId) const;

COM 接口参数：EB_CanEditMemberInfo([in] ULONGLONG nGroupCode, [in] ULONGLONG nMemberUserId, [out, retval] VARIANT_BOOL* pVal);

判断是否有编辑群组（部门）成员资料权限；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **eb::bigint nMemberUserId** | | 成员用户 ID |

## 2.7.5. EB_CanEditMyBaseMemberInfo

bool EB_CanEditMyBaseMemberInfo(eb::bigint nGroupCode) const;

COM 接口参数：EB_CanEditMyBaseMemberInfo([in] ULONGLONG nGroupCode, [out, retval] VARIANT_BOOL* pVal);

判断是否有编辑群组（部门）基础成员资料权限；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

## 2.7.6. EB_CanDeleteMemberInfo

bool EB_CanDeleteMemberInfo(eb::bigint nGroupCode,eb::bigint nMemberUserId) const;

COM 接口参数：EB_CanDeleteMemberInfo([in] ULONGLONG nGroupCode, [in] ULONGLONG nMemberUserId, [out, retval] VARIANT_BOOL* pVal);

判断是否有删除群组（部门）成员资料权限；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **eb::bigint nMemberUserId** | | 成员用户 ID |

## 2.7.7. EB_CanDeleteMemberInfo

bool EB_CanDeleteMemberInfo(eb::bigint nMemberCode) const;

COM 接口参数：EB_CanDeleteMemberInfo2([in] ULONGLONG nMemberCode, [out, retval] VARIANT_BOOL* pVal);

判断是否有删除群组（部门）成员资料权限；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nMemberCode** | | 群组（部门）成员编号 |

## 2.7.8. EB_CanEditGroupRes

bool EB_CanEditGroupRes(eb::bigint nGroupCode) const;

COM 接口参数：EB_CanEditGroupRes([in] ULONGLONG nGroupCode, [out, retval] VARIANT_BOOL* pVal);

判断是否有编辑和新建群组（部门）在线云盘资源权限；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

## 2.7.9. EB_CanDeleteGroupRes

bool EB_CanDeleteGroupRes(eb::bigint nGroupCode) const;

COM 接口参数：EB_CanDeleteGroupRes([in] ULONGLONG nGroupCode, [out, retval] VARIANT_BOOL* pVal);

判断是否有删除群组（部门）在线云盘资源权限；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

# 2.8. 在线云盘管理函数

## 2.8.1. EB_LoadRes

int EB_LoadRes(eb::bigint nGroupCode=0);

COM 接口：EB_LoadRes([in] ULONGLONG nGroupCode);

加载在线云盘信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | 0 | 群组（部门）编号<br>空为加载个人在线云盘 |

## 2.8.2. EB_GetDirAllRes

void EB_GetDirAllRes(eb::bigint nParentResId,std::vector<EB_ResourceInfo>& pOutResourceList) const;

COM 接口：EB_GetDirAllRes([in] ULONGLONG nParentResId, [out,retval] VARIANT* pOutResourceList);

获取指定目录下所有云资源；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nParentResId** | | 目录资源编号 |
| **std::vector<EB_ResourceInfo>& pOutResourceList** | | [输出]资源信息列表 |

## 2.8.3. EB_GetNotDirAllRes

void EB_GetNotDirAllRes(std::vector<EB_ResourceInfo>& pOutResourceList,eb::bigint nGroupCode=0) const;

COM 接口：EB_GetNotDirAllRes([in] ULONGLONG nGroupCode, [out,retval] VARIANT* pOutResourceList);

获取没有目录所有云资源；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **std::vector<EB_ResourceInfo>& pOutResourceList** | | [输出]资源信息列表 |
| **eb::bigint nGroupCode** | 0 | 群组（部门）编号<br>空为获取个人云资源 |

## 2.8.4. EB_HasSubRes

bool EB_HasSubRes(eb::bigint nParentResId) const;

COM 接口：EB_HasSubRes([in] ULONGLONG nParentResId, [out,retval] VARIANT_BOOL* pVal);

判断目录资源下，是否有资源信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nParentResId** | | 目录资源编号 |

## 2.8.5. EB_GetAllRes

void EB_GetAllRes(std::vector<EB_ResourceInfo>& pOutResourceList,eb::bigint nGroupCode=0) const;

COM 接口：EB_GetAllRes([in] ULONGLONG nGroupCode, [out,retval] VARIANT* pOutResourceList);

获取所有云资源信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **std::vector<EB_ResourceInfo>& pOutResourceList** | | [输出]资源信息列表 |
| **eb::bigint nGroupCode** | 0 | 群组（部门）编号<br>空为获取个人云资源 |

## 2.8.6. EB_GetAllFileRes

void EB_GetAllFileRes(std::vector<EB_ResourceInfo>& pOutResourceList,eb::bigint nGroupCode=0) const;

COM 接口：EB_GetAllFileRes([in] ULONGLONG nGroupCode, [out,retval] VARIANT* pOutResourceList);

获取所有云文件资源信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **std::vector<EB_ResourceInfo>& pOutResourceList** | | [输出]资源信息列表 |
| **eb::bigint nGroupCode** | 0 | 群组（部门）编号<br>空为获取个人云资源 |

## 2.8.7. EB_GetAllNoteRes

void EB_GetAllNoteRes(std::vector<EB_ResourceInfo>& pOutResourceList,eb::bigint nGroupCode=0) const;

COM 接口：EB_GetAllNoteRes([in] ULONGLONG nGroupCode, [out,retval] VARIANT* pOutResourceList);

获取所有云笔记资源信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **std::vector<EB_ResourceInfo>& pOutResourceList** | | [输出]资源信息列表 |
| **eb::bigint nGroupCode** | 0 | 群组（部门）编号<br>空为获取个人云资源 |

## 2.8.8. EB_AddDirRes

int EB_AddDirRes(const char* sDirName,eb::bigint nParentResId=0,eb::bigint nGroupCode=0);

COM 接口：EB_AddDirRes([in] BSTR sDirName, [in] ULONGLONG nParentResId, [in] ULONGLONG nGroupCode);

增加在线云目录资源；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sDirName** | | 目录名称 |
| **eb::bigint nParentResId** | 0 | 上线目录资源编号<br>空为增加根目录资源 |
| **eb::bigint nGroupCode** | 0 | 群组（部门）编号<br>空为增加个人目录资源 |

## 2.8.9. EB_AddFileRes

int EB_AddFileRes(const char* sFilePath,const char* sFileName,const char* sDescription,eb::bigint nParentResId=0,eb::bigint nGroupCode=0);

COM 接口 :EB_AddFileRes([in] BSTR sFilePath, [in] BSTR sFileName, [in] BSTR sDescription, [in] ULONGLONG nParentResId, [in] ULONGLONG nGroupCode);

增加在线云文件资源；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sFilePath** | | 本地文件保存路径 |
| **const char* sFileName** | | 文件名称 |
| **const char* sDescription** | | 文件描述 |
| **eb::bigint nParentResId** | 0 | 上线目录资源编号<br>空为增加根目录文件资源 |
| **eb::bigint nGroupCode** | 0 | 群组（部门）编号<br>空为增加个人文件资源 |

## 2.8.10. EB_AddNoteRes

int EB_AddNoteRes(const char* sNoteName,const char* sDescription,eb::bigint nParentResId=0,eb::bigint nGroupCode=0);

COM 接口：EB_AddNoteRes([in] BSTR sNoteName, [in] BSTR sDescription, [in] ULONGLONG nParentResId, [in] ULONGLONG nGroupCode);

增加在线云笔记资源；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sNoteName** | | 笔记标题名称 |
| **const char* sDescription** | | 笔记内容描述 |
| **eb::bigint nParentResId** | 0 | 上线目录资源编号<br>空为增加根目录笔记资源 |
| **eb::bigint nGroupCode** | 0 | 群组（部门）编号<br>空为增加个人笔记资源 |

## 2.8.11. EB_EditRes

int EB_EditRes(eb::bigint nResId,const char* sName,const char* sDescription);

COM 接口：EB_EditRes([in] ULONGLONG nResId, [in] BSTR sName, [in] BSTR sDescription);

修改在线云资源；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nResId** | | 在线云资源编号 |
| **const char* sName** | | 资源名称 |
| **const char* sDescription** | | 资源描述（内容） |

## 2.8.12. EB_MoveRes2Dir

int EB_MoveRes2Dir(eb::bigint nResId,eb::bigint nParentResId);

COM 接口：EB_MoveRes2Dir([in] ULONGLONG nResId, [in] ULONGLONG nParentResId);

移动在线云资源到指定目录下；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nResId** | | 在线云资源编号 |
| **eb::bigint nParentResId** | | 目录资源编号 |

## 2.8.13. EB_DeleteRes

int EB_DeleteRes(eb::bigint nResId);

COM 接口：EB_DeleteRes([in] ULONGLONG nResId);

删除在线云资源；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nResId** | | 在线云资源编号 |

## 2.8.14. EB_GetResourceInfo

bool EB_GetResourceInfo(eb::bigint nResId,EB_ResourceInfo* pOutResourceInfo);

COM 接口：EB_GetResourceInfo([in] ULONGLONG nResId, [out,retval] IEB_ResourceInfo** pVal);

获取云资源信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nResId** | | 在线云资源编号 |

| | |
|---|---|
| **EB_ResourceInfo* pOutResourceInfo** | [输出]云资源信息 |

## 2.8.15. EB_GetResourceCmInfo

bool EB_GetResourceCmInfo(eb::bigint nResId,std::string& pOutResourceInfo);

COM 接口：EB_GetResourceCmInfo([in] ULONGLONG nResId, [out,retval] BSTR* pVal);

获取云资源 CM 信息；该 CM 信息，用于发送聊天信息，如表情发送等；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nResId** | | 在线云资源编号 |
| **std::string& pOutResourceInfo** | | [输出]云资源 CM 信息 |

## 2.8.16. EB_DownloadFileRes

int EB_DownloadFileRes(eb::bigint nResId,const char * sSaveTo);

COM 接口：EB_DownloadFileRes([in] ULONGLONG nResId, [in] BSTR sSaveTo);

下载在线云资源文件，并保存到指定路径；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nResId** | | 在线云资源编号 |
| **const char * sSaveTo** | | 下载资源保存路径 |

## 2.8.17. EB_CancelFileRes

int EB_CancelFileRes(eb::bigint nResId,eb::bigint nMsgId);

COM 接口：EB_CancelFileRes([in] ULONGLONG nResId, [in] ULONGLONG nMsgId);

取消正在下载、或上传云资源文件；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nResId** | | 在线云资源编号 |
| **eb::bigint nMsgId** | | 资源文件传输消息编号 |

# 2.9. 聊天会话管理函数

## 2.9.1. EB_CallMember

int EB_CallMember(eb::bigint nMemberCode,eb::bigint nExistCallId=0);

COM 接口 :EB_CallMember([in] ULONGLONG nMemberCode, [in] ULONGLONG nExistCallId);

呼叫群组（部门）成员用户，发起一个聊天请求；或邀请用户进入当前会话；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nMemberCode** | 0 | 群组（部门）成员编号<br>非成员编号留空 |
| **eb::bigint nExistCallId** | 0 | 存在会话<br>空为新会话；<br>不为空用于邀请对方多人会话 |

## 2.9.2. EB_CallAccount

int EB_CallAccount(const char * sToAccount,eb::bigint nExistCallId=0);

COM 接口：EB_CallAccount([in] BSTR sToAccount, [in] ULONGLONG nExistCallId);

呼叫用户，发起一个聊天请求；或邀请用户进入当前会话；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char * sToAccount** | | 呼叫对方邮箱帐号 |
| **eb::bigint nExistCallId** | 0 | 存在会话<br>空为新会话；<br>不为空用于邀请对方多人会话 |

## 2.9.3. EB_CallUserId

int EB_CallUserId(eb::bigint nToUserId,eb::bigint nExistCallId=0);

COM 接口：EB_CallUserId([in] ULONGLONG nToUserId, [in] ULONGLONG nExistCallId);

呼叫用户，发起一个聊天请求；或邀请用户进入当前会话；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nToUserId** | | 呼叫对方用户 ID |

| | | |
|---|---|---|
| **eb::bigint nExistCallId** | 0 | 存在会话<br>空为新会话；<br>不为空用于邀请对方多人会话 |

## 2.9.4. EB_Call2Group

int EB_Call2Group(eb::bigint nCallId,const char* sToAccount);

COM 接口：EB_Call2Group([in] ULONGLONG nCallId, [in] BSTR sToAccount);

一对一会话转换临时讨论组，同时邀请第三方成员进入讨论组；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nCallId** | | 一对一会话编号 |
| **const char* sToAccount** | | 邀请第三方帐号，加入讨论组 |

## 2.9.5. EB_CallGroup

int EB_CallGroup(eb::bigint nGroupCode);

COM 接口：EB_CallGroup([in] ULONGLONG nGroupCode);

发起群组（部门）聊天会话；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

## 2.9.6. EB_CallAnswer

int EB_CallAnswer(eb::bigint nCallId,bool bAccept);

COM 接口：EB_CallAnswer([in] ULONGLONG nCallId, [in] VARIANT_BOOL bAccept);

接受或拒绝一个聊天会话请求，同企业，同群组成员会自动接收响应，该函数用于外部用户呼叫；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nCallId** | | 会话编号 |
| **bool bAccept** | | 是否接通该会话请求 true/false |

### 2.9.7. EB_CallHangup

int EB_CallHangup(eb::bigint nCallId);

COM 接口：EB_CallHangup([in] ULONGLONG nCallId);

挂断会话；主要用于挂断外部用户会话，挂断会话后，对方需要重新呼叫，本端接通会话后才能正常通知；

如果不想挂断会话，使用 EB_CallExit；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |

### 2.9.8. EB_CallExit

void EB_CallExit(eb::bigint nCallId);

COM 接口：EB_CallExit([in] ULONGLONG nCallId);

退出会话；用于上层退出聊天界面，清空会话信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |

### 2.9.9. EB_RichBufferAddText

void EB_RichBufferAddText(eb::bigint nCallId,const char* sText);

COM 接口：EB_RichBufferAddText([in] ULONGLONG nCallId, [in] BSTR sText);

会话聊天 RICH 格式缓存添加一个文本；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **const char * sText** | | 文本信息 |

### 2.9.10. EB_RichBufferAddObject

void EB_RichBufferAddObject(eb::bigint nCallId,const char* pData,unsigned long nSize);

COM 接口 1：EB_RichBufferAddObject([in] ULONGLONG nCallId, [in] BYTE* pData, [in]ULONG nSize);

COM 接口 2：EB_RichBufferAddObject2([in] ULONGLONG nCallId, [in] VARIANT* pData, [in]ULONG nSize);

会话聊天 RICH 格式缓存添加一个图片对象（如 JPG）；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **const char * pData** | | 对象数据 |
| **unsigned long nSize** | | 数据大小 |

## 2.9.11. EB_RichBufferAddResource

void EB_RichBufferAddResource(eb::bigint nCallId,const char* sResource);

COM 接口：EB_RichBufferAddResource([in] ULONGLONG nCallId, [in] BSTR sResource);

会话聊天 RICH 格式缓存添加一个资源对象（如表情）；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **const char * sResource** | | 资源信息 |

## 2.9.12. EB_RichBufferClear

void EB_RichBufferClear(eb::bigint nCallId);

COM 接口：EB_RichBufferClear([in] ULONGLONG nCallId);

清空会话聊天 RICH 格式缓存；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |

## 2.9.13. EB_SendRichBuffer

int EB_SendRichBuffer(eb::bigint nCallId,eb::bigint nToUserId=0,bool bPrivate=false);

COM 接口：EB_SendRichBuffer([in] ULONGLONG nCallId, [in] ULONGLONG nToUserId, [in] VARIANT_BOOL bPrivate);

发送会话聊天 RICH 格式缓存内容，发送完成自动清空缓存；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **eb::bigint nToUserId** | 0 | 只发给指定用户<br>用于群组（部门）会话中<br>一对一会话留空 |
| **bool bPrivate** | false | 是否私聊 true/false<br>用于群组（部门）会话中，群组会话，选择只发给某用户私聊，其他会话成员将接收不到；<br>一对一会话默认不填写 |

## 2.9.14. EB_SendText

int EB_SendText(eb::bigint nCallId,const EB_char* sTextMsg,eb::bigint nToUserId=0,bool bPrivate=false);

COM 接口：EB_SendText([in] ULONGLONG nCallId, [in] BSTR sTextMsg, [in] ULONGLONG nToUserId, [in] VARIANT_BOOL bPrivate);

发送文本信息等；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **const char * sTextMsg** | | 文本信息 |
| **eb::bigint nToUserId** | 0 | 用于群组（部门）会话中<br>只发给群组内指定用户<br>一对一会话留空 |
| **bool bPrivate** | false | 是否私聊 true/false<br>用于群组（部门）会话中，群组会话，选择只发给某用户私聊，其他会话成员将接收不到；<br>一对一会话默认不填写 |

## 2.9.15. EB_SendRich

int EB_SendRich(eb::bigint nCallId,const EB_ChatRoomRichMsg* pRichMsg,eb::bigint nToUserId=0,bool bPrivate=false);

COM 接口：EB_SendRich([in] ULONGLONG nCallId, [in] VARIANT* pRichMsg, [in] ULONGLONG nToUserId, [in] VARIANT_BOOL bPrivate);

发送 RICH 富格式信息，包括文本、图片、表情资源等；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **const EB_ChatRoomRichMsg* pRichMsg** | | 富格式结构信息 |
| **eb::bigint nToUserId** | 0 | 用于群组（部门）会话中<br>只发给群组内指定用户<br>一对一会话留空 |
| **bool bPrivate** | false | 是否私聊 true/false<br>用于群组（部门）会话中，群组会话，选择只发给某用户私聊，其他会话成员将接收不到；<br>一对一会话默认不填写 |

## 2.9.16. EB_SendFile

int EB_SendFile(eb::bigint nCallId,const char* sFilePath,eb::bigint nToUserId=0,bool bPrivate=false,bool bOffFile=false);

COM 接口：EB_SendFile([in] ULONGLONG nCallId, [in] BSTR sFilePath, [in] ULONGLONG nToUserId, [in] VARIANT_BOOL bPrivate, [in] VARIANT_BOOL bOffFile);

发送文件，文件需要等待对方接收才开始传送；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **const char* sFilePath** | | 文件路径 |
| **eb::bigint nToUserId** | 0 | 只发给指定用户<br>用于群组（部门）会话中<br>一对一会话留空 |
| **bool bPrivate** | false | 是否私聊 true/false<br>用于群组（部门）会话中，群组会话，选择只发给某用户私聊，其他会话成员将接收不到；<br>一对一会话默认不填写 |
| **bool bOffFile** | false | 是否发送离线文件 true/false |

## 2.9.17. EB_AcceptFile

int EB_AcceptFile(eb::bigint nCallId,eb::bigint nMsgId,const char * sSaveTo);

COM 接口：EB_AcceptFile([in] ULONGLONG nCallId, [in] ULONGLONG nMsgId, [in] BSTR sSaveTo);

接收文件；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **eb::bigint nMsgId** | | 文件消息编号 |
| **const char * sSaveTo** | | 文件保存路径 |

## 2.9.18. EB_Save2CloudDrive

int EB_Save2CloudDrive(eb::bigint nCallId,eb::bigint nMsgId);

COM 接口：EB_Save2CloudDrive([in] ULONGLONG nCallId, [in] ULONGLONG nMsgId);

接收文件；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **eb::bigint nMsgId** | | 离线文件消息编号 |

## 2.9.19. EB_CancelFile

int EB_CancelFile(eb::bigint nCallId,eb::bigint nMsgId);

COM 接口：EB_CancelFile([in] ULONGLONG nCallId, [in] ULONGLONG nMsgId);

取消或拒绝接收文件；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **eb::bigint nMsgId** | | 文件消息编号 |

## 2.9.20. EB_GetCallAccountInfoList

bool EB_GetCallAccountInfoList(eb::bigint nCallId,std::vector<EB_AccountInfo>& pOutUserList) const;

COM 接口：EB_GetCallAccountInfoList([in] ULONGLONG nCallId, [out,retval] VARIANT* pVal);

获取会话成员帐号信息列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **std::vector<EB_AccountInfo>& pOutUserList** | | [输出]会话成员信息列表 |

## 2.9.21. EB_GetCallUserIdList

bool EB_GetCallUserList(eb::bigint nCallId,std::vector<eb::bigint>& pOutUserList) const;

COM 接口：EB_GetCallUserIdList([in] ULONGLONG nCallId, [out,retval] VARIANT* pVal);

获取会话成员帐号列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **std::vector<eb::bigint>& pOutUserList** | | [输出]会话成员用户 ID 列表 |

## 2.9.22. EB_GetCallAccountList

bool EB_GetCallAccountList(eb::bigint nCallId,std::vector<std::string>& pOutUserList) const;

COM 接口：EB_GetCallAccountList([in] ULONGLONG nCallId, [out,retval] VARIANT* pVal);

获取会话成员邮箱帐号列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **std::vector<std::string>& pOutUserList** | | [输出]会话成员帐号列表 |

## 2.9.23. EB_GetCallAccountInfo

bool EB_GetCallAccountInfo(eb::bigint nCallId,eb::bigint nUserId,EB_AccountInfo* pOutAccountInfo) const;

COM 接口：EB_GetCallAccountInfo([in] ULONGLONG nCallId, [in] ULONGLONG nUserId, [out,retval] IEB_AccountInfo** pVal);

获取会话成员信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |

| | |
|---|---|
| **eb::bigint nCallId** | 会话编号 |
| **eb::bigint nUserId** | 会话成员帐号 |
| **EB_AccountInfo* pOutAccountInfo** | [输出]会话成员信息 |

# 2.10. 联系人管理函数

## 2.10.1. 描述

通讯录（联系人）资料类似于手机通讯录，主要用于资料保存和方便呼叫；

通讯录用户没有上下线状态事件通知；

## 2.10.2. EB_LoadContact

int EB_LoadContact(void);

COM 接口：EB_LoadContact(void);

加载我的在线通讯录（联系人）资料；登录成功后，该函数调用一次；

## 2.10.3. EB_EditUGInfo

int EB_EditUGInfo(eb::bigint nUGId, const char* sGroupName);

COM 接口：EB_EditUGInfo([in] LONGLONG nUGId, [in] BSTR sGroupName);

修改或新建一条分组资料；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nUGId** | | 分组 ID，0 表示新建 |
| **const char* sGroupName** | | 分组名称，不能为空 |

## 2.10.4. EB_DeleteUGInfo

int EB_DeleteUGInfo(eb::bigint nUGId);

COM 接口：EB_DeleteUGInfo([in] LONGLONG nUGId);

删除一条分组资料，删除成功，该分组下所有联系人移到默认分组；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nUGId** | | 分组 ID，不能为 0； |

## 2.10.5. EB_GetUGInfoList

void EB_GetUGInfoList(std::vector<EB_UGInfo>& pOutUGInfoList) const;

COM 接口：EB_GetUGInfoList([out,retval] VARIANT* pVal);

获取分组信息列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **std::vector<EB_UGInfo>& pOutUGInfoList** | | [输出]分组信息列表 |

## 2.10.6. EB_ReqAddContact

int EB_ReqAddContact(eb::bigint nContactUserId, const char* sDescription);

COM 接口：EB_ReqAdContact([in] LONGLONG nContactUserId, [in] BSTR sDescription);

申请添加联系人；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nContactUserId** | | 联系人用户 ID，不能为 0； |
| **const char* sDescription** | | 添加验证信息 |

## 2.10.7. EB_EditContact

int EB_EditContact(const EB_ContactInfo* pContactInfo);

COM 接口：EB_EditContact([in] IDispatch* pVal);

修改或新建一条联系人资料；

| 参数 | | 默认值（非空） | 描述 |
| --- | --- | --- | --- |
| **const EB_ContactInfo* pContactInfo** | | | 联系人信息<br>联系人信息不存在，将自动保存 |

### 2.10.8. EB_DeleteContact

int EB_DeleteContact(const char* sContactAccount);

COM 接口：EB_DeleteContact([in] BSTR sContactAccount);

删除一条联系人资料；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sContactAccount** | | 联系人帐号（邮箱） |

### 2.10.9. EB_GetContactList

void EB_GetContactList(std::vector<EB_ContactInfo>& pOutContactList) const;

COM 接口：EB_GetContactList([out,retval] VARIANT* pVal);

获取联系人信息列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **std::vector<EB_ContactInfo>& pOutContactList** | | [输出]联系人信息列表 |

### 2.10.10. EB_GetContactInfo

bool EB_GetContactInfo(const char* sContactAccount,EB_ContactInfo* pOutContactInfo) const;

COM 接口：EB_GetContactInfo([in] BSTR sContactAccount, [out,retval] IEB_ContactInfo** pVal);

获取联系人信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sContactAccount** | | 联系人帐号（邮箱） |
| **EB_ContactInfo* pOutContactInfo** | | [输出]联系人信息 |

### 2.10.11. EB_IsMyContactAccount

bool EB_IsMyContactAccount(const char* sContactAccount) const;

COM 接口：EB_IsMyContactAccount([in] BSTR sContactAccount, [out,retval] VARIANT_BOOL* pVal);

判断是否我的联系人帐号；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const char* sContactAccount** | | 联系人帐号（邮箱） |

# 2.11. 企业组织结构（个人群组）管理函数

## 2.11.1. 描述

群分四种类型：企业内部的部门、项目组，以及个人群组和临时讨论组；

同企业，同群（包括部门、项目组、个人群组、临时讨论组）成员之间，可以接收用户上下线状态事件通知；

## 2.11.2. EB_LoadOrg

int EB_LoadOrg(void);

COM 接口：EB_LoadOrg(void);

加载我的企业组织结构信息和个人群组等数据；登录成功后，该函数调用一次；

## 2.11.3. EB_LoadGroup

int EB_LoadGroup(eb::bigint nGroupCode, bool bLoadMember);

COM 接口：EB_LoadGroup([in] ULONGLONG nGroupCode, [in] VARIANT_BOOL bLoadMember);

申请加入群组（部门）；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **bool bLoadMember** | | 是否加载部门成员<br>false：只加载部门资料 |

| | |
|---|---|
| | true：同时加载部门成员 |

## 2.11.4. EB_GetEnterpriseInfo

bool EB_GetEnterpriseInfo(EB_EnterpriseInfo* pOutEnterpriseInfo,eb::bigint nEnterpriseCode=0) const;

COM 接口：EB_GetEnterpriseInfo([in] ULONGLONG nEnterpriseCode, [out,retval] IEB_EnterpriseInfo** pVal);

获取企业资料信息；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **EB_EnterpriseInfo* pOutEnterpriseInfo** | | [输出]企业资料信息 |
| **eb::bigint nEnterpriseCode** | 0 | 企业代码，空为获取当前默认企业 |

## 2.11.5. EB_IsEnterpriseUser

bool EB_IsEnterpriseUser(eb::bigint nEnterpriseCode=0) const;

COM 接口：EB_IsEnterpriseUser([in] ULONGLONG nEnterpriseCode, [out,retval] VARIANT_BOOL* pVal);

判断登录用户是否企业员工。

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nEnterpriseCode** | 0 | 企业代码，空为获取当前默认企业 |

## 2.11.6. EB_GetEnterpriseName

bool EB_GetEnterpriseName(std::string& pOutEnterpriseName,eb::bigint nEnterpriseCode=0) const;

COM 接口：EB_GetEnterpriseName([in] ULONGLONG nEnterpriseCode, [out,retval] BSTR* pVal);

获取企业资料信息；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **std::string& pOutEnterpriseName** | | [输出]企业名称 |

| | | |
|---|---|---|
| **eb::bigint nEnterpriseCode** | 0 | 企业代码，空为获取当前默认企业 |

## 2.11.7. EB_GetEnterpriseMemberSize

void EB_GetEnterpriseMemberSize(eb::bigint nEnterpriseCode, int& pOutMemberSize,int& pOutOnlineSize) const;

COM 接口：EB_GetEnterpriseMemberSize([in] ULONGLONG nEnterpriseCode, [out] LONG* pOutMemberSize, [out] LONG *pOutOnlineSize);

获取企业资料信息；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nEnterpriseCode** | | 企业代码 |
| **int& pOutMemberSize** | | [输出]企业员工总数 |
| **int& pOutOnlineSize** | | [输出]企业在线员工总数 |

## 2.11.8. EB_EditEnterprise

int EB_EditEnterprise(const EB_EnterpriseInfo* pEnterpriseInfo);

COM 接口：EB_EditEnterprise([in] IDispatch* newVal);

修改企业资料信息；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **const EB_EnterpriseInfo* pEnterpriseInfo** | | 企业资料信息 |

## 2.11.9. EB_EditGroup

int EB_EditGroup(const EB_GroupInfo* pGroupInfo);

COM 接口：EB_EditGroup([in] IDispatch* newVal);

修改或新建群组（部门）资料信息；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **const EB_GroupInfo* pGroupInfo** | | 群组（部门）资料信息<br>pGroupInfo->m_sGroupCode 为空，表示新建 |

## 2.11.10. EB_DeleteGroup

int EB_DeleteGroup(eb::bigint nGroupCode);

COM 接口：EB_DeleteGroup([in] ULONGLONG nGroupCode);

删除部门或解散群组；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

## 2.11.11. EB_JoinGroup

int EB_JoinGroup(eb::bigint nGroupCode);

COM 接口：EB_JoinGroup([in] ULONGLONG nGroupCode);

申请加入群组（部门）；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

## 2.11.12. EB_ExitGroup

int EB_ExitGroup(eb::bigint nGroupCode);

COM 接口：EB_ExitGroup([in] ULONGLONG nGroupCode);

退出个人群组或临时讨论组；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组编号 |

## 2.11.13. EB_SetMyGroupHeadFile

int EB_SetMyGroupHeadFile(eb::bigint nGroupCode,const char* sImagePath);

COM 接口：EB_SetMyGroupHeadFile([in] ULONGLONG nGroupCode, [in] BSTR sImagePath);

设置个人群组（部门）头像；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |

| | |
|---|---|
| **eb::bigint nGroupCode** | 群组（部门）编号 |
| **const char* sImagePath** | 头像文件路径 |

## 2.11.14. EB_SetMyGroupHeadRes

int EB_SetMyGroupHeadRes(eb::bigint nGroupCode,eb::bigint nResId);

COM 接口：EB_SetMyGroupHeadRes([in] ULONGLONG nGroupCode, [in] ULONGLONG nResId);

设置个人群组（部门）头像；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **eb::bigint nResId** | | 头像资源 ID |

## 2.11.15. EB_GetGroupInfo

bool EB_GetGroupInfo(eb::bigint nGroupCode,EB_GroupInfo* pOutGroupInfo) const;

COM 接口：EB_GetGroupInfo([in] ULONGLONG nGroupCode, [out,retval] IEB_GroupInfo** pVal);

获取群组（部门）信息；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **EB_GroupInfo* pOutGroupInfo** | | [输出]群组（部门）信息 |

## 2.11.16. EB_GetGroupName

bool EB_GetGroupName(eb::bigint nGroupCode,std::string& pOutGroupName) const;

COM 接口：EB_GetGroupName([in] ULONGLONG nGroupCode, [out,retval] BSTR* pVal);

获取群组（部门）名称；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **std::string& pOutGroupName** | | [输出]群组（部门）名称 |

## 2.11.17. EB_GetGroupCreator

bool EB_GetGroupCreator(eb::bigint nGroupCode,eb::bigint& pOutGroupCreator) const;

COM 接口：EB_GetGroupCreator([in] ULONGLONG nGroupCode, [out,retval] ULONGLONG* pVal);

获取群组（部门）创建者；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **eb::bigint& pOutGroupCreator** | | [输出]群组（部门）创建者用户 ID |

## 2.11.18. EB_GetGroupType

bool EB_GetGroupType(eb::bigint nGroupCode,EB_GROUP_TYPE& pOutGroupType) const;

COM 接口：EB_GetGroupType([in] ULONGLONG nGroupCode, [out,retval] SHORT* pVal);

获取群组（部门）类型；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **EB_GROUP_TYPE& pOutGroupType** | | [输出]群组（部门）类型 |

## 2.11.19. EB_GetGroupMemberInfoList

bool EB_GetGroupMemberInfoList(eb::bigint nGroupCode,std::vector<EB_MemberInfo>& pOutMemberInfoList) const;

COM 接口：EB_GetGroupMemberInfoList([in] ULONGLONG nGroupCode, [out,retval] VARIANT* pVal);

获取群组（部门）成员信息列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **std::vector<EB_MemberInfo>& pOutMemberInfoList** | | [输出]群组（部门）成员信息列表 |

## 2.11.20. EB_GetGroupMemberCodeList

bool EB_GetGroupMemberCodeList(eb::bigint nGroupCode,std::vector<eb::bigint>& pOutMemberCodeList) const;

COM 接口：EB_GetGroupMemberCodeList([in] ULONGLONG nGroupCode, [out,retval] VARIANT* pVal);

获取群组（部门）成员用户 ID 列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **std::vector<eb::bigint>& pOutMemberCodeList** | | [输出]群组（部门）成员编号列表 |

## 2.11.21. EB_GetGroupMemberUserIdList

bool EB_GetGroupMemberUserIdList(eb::bigint nGroupCode,std::vector<eb::bigint>& pOutMemberUserIdList) const;

COM 接口：EB_GetGroupMemberUserIdList([in] ULONGLONG nGroupCode, [out,retval] VARIANT* pVal);

获取群组（部门）成员用户 ID 列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **std::vector<eb::bigint>& pOutMemberUserIdList** | | [输出]群组（部门）成员用户 ID 列表 |

## 2.11.22. EB_GetGroupMemberAccountList

bool EB_GetGroupMemberAccountList(eb::bigint nGroupCode,std::vector<std::string>& pOutMemberAccountList) const;

COM 接口：EB_GetGroupMemberAccountList([in] ULONGLONG nGroupCode, [out,retval] VARIANT* pVal);

获取群组（部门）成员邮箱帐号列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

| | |
|---|---|
| **std::vector<std::string>& pOutMemberInfoList** | [输出]群组（部门）成员邮箱帐号列表 |

## 2.11.23. EB_GetGroupMemberSize

int EB_GetGroupMemberSize(eb::bigint nGroupCode) const;

COM 接口：EB_GetGroupMemberSize([in] ULONGLONG nGroupCode, [out,retval] ULONG* pVal);

返回群组（部门）成员数量；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

## 2.11.24. EB_GetGroupOnlineSize

int EB_GetGroupOnlineSize(eb::bigint nGroupCode) const;

COM 接口：EB_GetGroupOnlineSize([in] ULONGLONG nGroupCode, [out,retval] ULONG* pVal);

返回群组（部门）在线成员数量；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

## 2.11.25. EB_IsMyGroup

bool EB_IsMyGroup(eb::bigint nGroupCode) const;

COM 接口：EB_IsMyGroup([in] ULONGLONG nGroupCode, [out,retval] VARIANT_BOOL* pVal);

判断是否是我的群组（部门）；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |

## 2.11.26. EB_EditMember

int EB_EditMember(const EB_MemberInfo* pMemberInfo);

COM 接口：EB_EditMember([in] IDispatch* newVal);

修改或新建群组（部门）成员信息，主要用于添加部门员工或修改员工资料；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **const EB_MemberInfo* pMemberInfo** | | 群组（部门）成员信息 pMemberInfo->m_sMemberCode 成员编号为空，表示新建 |

## 2.11.27. EB_DeleteMember

int EB_DeleteMember(eb::bigint nMemberCode, bool bDeleteAccount=true);

COM 接口：EB_DeleteMember([in] ULONGLONG nMemberCode, [in] VARIANT_BOOL bDeleteAccount);

删除群组（部门）成员；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nMemberCode** | | 群组（部门）成员编号 |
| **bool bDeleteAccount** | true | 是否删除登录帐号 true/false |

## 2.11.28. EB_GetMemberInfoByUserId

bool EB_GetMemberInfoByUserId(EB_MemberInfo* pOutMemberInfo,eb::bigint nGroupCode,eb::bigint nMemberUserId) const;

COM 接口：EB_GetMemberInfoByUserId([in] ULONGLONG nGroupCode, [in] ULONGLONG nMemberUserId, [out,retval] IEB_MemberInfo** pVal);

获取群组（部门）成员信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **EB_MemberInfo* pOutMemberInfo** | | [输出]群组（部门）成员信息 |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **eb::bigint nMemberUserId** | | 群组（部门）成员用户 ID |

## 2.11.29. EB_GetMemberInfoByAccount

bool EB_GetMemberInfoByAccount(EB_MemberInfo* pOutMemberInfo,eb::bigint nGroupCode,const char * sMemberAccount) const;

COM 接口：EB_GetMemberInfoByAccount([in] ULONGLONG nGroupCode, [in] BSTR sMemberAccount, [out,retval] IEB_MemberInfo** pVal);

获取群组（部门）成员信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **EB_MemberInfo* pOutMemberInfo** | | [输出]群组（部门）成员信息 |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **const char * sMemberAccount** | | 群组（部门）成员邮箱帐号 |

## 2.11.30. EB_GetMemberInfoByMemberCode

bool EB_GetMemberInfoByMemberCode(EB_MemberInfo* pOutMemberInfo,eb::bigint nMemberCode) const;

COM 接口 :EB_GetMemberInfoByMemberCode([in] ULONGLONG nMemberCode, [out,retval] IEB_MemberInfo** pVal);

获取群组（部门）成员信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **EB_MemberInfo* pOutMemberInfo** | | [输出]群组（部门）成员信息 |
| **eb::bigint nMemberCode** | | 群组（部门）成员编号 |

## 2.11.31. EB_GetMemberNameByUserId

bool EB_GetMemberNameByUserId (eb::bigint nGroupCode,eb::bigint nMemberUserId,std::string& pOutMemberName) const;

COM 接口 : EB_GetMemberNameByUserId([in] ULONGLONG nGroupCode , [in] ULONGLONG nMemberUserId, [out,retval] BSTR* pVal);

获取群组（部门）成员名称；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **eb::bigint nMemberUserId** | | 群组（部门）成员帐号 |
| **std::string& pOutMemberName** | | [输出]群组（部门）成员名称 |

## 2.11.32. EB_GetMemberNameByMemberCode

bool EB_GetMemberNameByMemberCode(eb::bigint nMemberCode,std::string& pOutMemberName) const;

COM 接口：EB_GetMemberNameByMemberCode([in] ULONGLONG nMemberCode, [out,retval] BSTR* pVal);

获取群组（部门）成员名称；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nMemberCode** | | 群组（部门）成员编号 |
| **std::string& pOutMemberName** | | [输出]群组（部门）成员名称 |

## 2.11.33. EB_GetMemberLineState

bool EB_GetMemberLineState(eb::bigint nMemberCode,EB_USER_LINE_STATE& pOutLineState) const;

COM 接口：EB_GetMemberLineState([in] ULONGLONG nMemberCode, [out,retval] SHORT* pVal);

获取群组（部门）成员名称；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nMemberCode** | | 群组（部门）成员编号 |
| **EB_USER_LINE_STATE& pOutLineState** | | [输出]群组（部门）成员在线状态 |

## 2.11.34. EB_GetMyMemberInfo

bool EB_GetMyMemberInfo(EB_MemberInfo* pOutMemberInfo,eb::bigint nGroupCode=0) const;

COM 接口：EB_GetMyMemberInfo([in] ULONGLONG nGroupCode, [out,retval] IEB_MemberInfo** pVal);

获取我的群组（部门）成员信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **EB_MemberInfo*** | | [输出]群组（部门）成员信息 |

| pOutMemberInfo | | |
|---|---|---|
| **eb::bigint nGroupCode** | 0 | 群组（部门）成员编号<br>0 为获取默认群成员信息 |

## 2.11.35. EB_GetMyMemberList

void EB_GetMyMemberList(std::vector<EB_MemberInfo>& pOutMemberInfoList) const;

COM 接口：EB_GetMyMemberList([out,retval] VARIANT* pVal);

获取我的所有群组（部门）成员信息列表；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **std::vector<EB_MemberInfo>& pOutMemberInfoList** | | [输出]群组（部门）成员信息列表 |

## 2.11.36. EB_IsExistMemberByUserId

bool EB_IsExistMemberByUserId(eb::bigint nGroupCode,eb::bigint nMemberUserId)const;

COM 接口：EB_IsExistMemberByUserId([in] ULONGLONG nGroupCode, [in] ULONGLONG nMemberUserId, [out,retval] VARIANT_BOOL* pVal);

判断是否存在群组（部门）成员；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **eb::bigint nMemberUserId** | | 群组（部门）成员帐号 |

## 2.11.37. EB_IsExistMemberByAccount

bool EB_IsExistMemberByAccount(eb::bigint nGroupCode,const char* sMemberAccount)const;

COM 接口：EB_IsExistMemberByAccount([in] ULONGLONG nGroupCode, [in] BSTR sMemberAccount, [out,retval] VARIANT_BOOL* pVal);

判断是否存在群组（部门）成员；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nGroupCode** | | 群组（部门）编号 |
| **const char* sMemberAccount** | | 群组（部门）成员帐号 |

### 2.11.38. EB_IsExistMemberByMemberCode

bool EB_IsExistMemberByMemberCode(eb::bigint nMemberCode) const;

COM 接口：EB_IsExistMemberByMemberCode([in] ULONGLONG nMemberCode, [out,retval] VARIANT_BOOL* pVal);

判断是否存在群组（部门）成员；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nMemberCode** | | 群组（部门）成员编号 |

## 2.12. 表情信息管理函数

### 2.12.1. 描述

企业可以定制内部专属表情库，详细信息请联系恩布公司；

### 2.12.2. EB_GetMyEmotionList

bool EB_GetMyEmotionList(std::vector<EB_EmotionInfo>& pOutEmotionList) const;

COM 接口：EB_GetMyEmotionList([out,retval] VARIANT* pVal);

获取我的表情资源列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **std::vector<EB_EmotionInfo>& pOutEmotionList** | | [输出]表情资源信息列表 |

### 2.12.3. EB_GetDefaultHeadList

bool EB_GetDefaultHeadList(std::vector<EB_EmotionInfo>& pOutEmotionList) const;

COM 接口：EB_GetDefaultHeadList([out,retval] VARIANT* pVal);

获取系统默认头像资源列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **std::vector<EB_EmotionInfo>&** | | [输出]头像资源信息列表 |

| pOutEmotionList |
|---|

# 2.13. 搜索查找函数

## 2.13.1. EB_FindAllGroupInfo

void EB_FindAllGroupInfo(CEBSearchCallback * pCallback,eb::bigint nEnterpriseCode=0,unsigned long dwParam=0);

COM 接口：EB_FindAllGroupInfo([in] IDispatch* pConnection, [in] ULONGLONG nEnterpriseCode, [in] ULONG ulParam);

查找企业所有群组（部门）信息；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **CEBSearchCallback * pCallback** | | 搜索查找结果回调类 |
| **eb::bigint nEnterpriseCode** | 0 | 企业代码<br>空为搜索查找所有，包括个人群组 |
| **unsigned long dwParam** | 0 | 回调参数，由 pCallback 返回 |

## 2.13.2. EB_FindAllContactInfo

void EB_FindAllContactInfo(CEBSearchCallback * pCallback,unsigned long dwParam=0);

COM 接口：EB_FindAllContactInfo([in] IDispatch* pConnection, [in] ULONG ulParam);

查找所有所有通讯录（联系人）信息；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **CEBSearchCallback * pCallback** | | 搜索查找结果回调类 |
| **unsigned long dwParam** | 0 | 回调参数，由 pCallback 返回 |

## 2.13.3. EB_SearchUserInfo

void EB_SearchUserInfo(CEBSearchCallback * pCallback,const char* sSearchKey,unsigned long dwParam=0);

COM 接口：EB_SearchUserInfo([in] IDispatch* pConnection, [in] BSTR sSearchKey, [in] ULONG ulParam);

查找用户，包括企业部门员工，个人群组和通讯录等所有用户；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **CEBSearchCallback * pCallback** | | 搜索查找结果回调类 |
| **const char* sSearchKey** | | 搜索关键词 |
| **unsigned long dwParam** | 0 | 回调参数，由 pCallback 返回 |

# 2.14. 视频通话函数

## 2.14.1. 描述

恩布平台支持一对一视频通话，也支持多人视频会议功能。

## 2.14.2. EB_GetVideoDeviceList

static void EB_GetVideoDeviceList(std::vector<std::string>& pOutVideoDeviceList);

COM 接口：EB_GetVideoDeviceList([out,retval] VARIANT* pVal);

获取本地视频设备名称列表；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **std::vector<std::string>& pOutVideoDeviceList** | | [输出]视频设备名称列表 |

## 2.14.3. EB_GetDefaultVideoMediaType

static int EB_GetDefaultVideoMediaType(void);

COM 接口：EB_GetDefaultVideoMediaType([out,retval] LONG* pVal);

返回默认视频媒体类型；

## 2.14.4. EB_SetVideoCallback

void EB_SetVideoCallback(eb::bigint nCallId, PVideoDataCallBack cb, DWORD dwParam);

COM 接口：EB_SetVideoCallback([in] ULONGLONG nCallId, [in] IDispatch* pVideoDataCallback, [in] ULONG nUserData);

设置会话视频显示数据统一回调函数；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **PVideoDataCallBack cb** | | 视频数据回调地址 |
| **DWORD dwParam** | | 自定义回调参数 |

## 2.14.5. EB_SetVideoHwnd

void EB_SetVideoHwnd(eb::bigint nCallId,HWND pHwnd);

COM 接口：该接口使用 COM 事件连接点；

设置会话视频业务回调窗口句柄；空为取消，使用默认句柄 EB_SetMsgHwnd；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **HWND pHwnd** | | 视频业务回调窗口句柄 |

## 2.14.6. EB_VideoRequest

int EB_VideoRequest(eb::bigint nCallId,EB_VIDEO_TYPE nVideoType=EB_VIDEO_BOTH);

COM 接口：EB_VideoRequest([in] ULONGLONG nCallId, [in] SHORT nVideoType);

请求会话视频通讯；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **EB_VIDEO_TYPE nVideoType** | EB_VIDEO_BOTH | 视频通讯类型<br>语音或视频 |

## 2.14.7. EB_VideoAck

int EB_VideoAck(eb::bigint nCallId,bool bAccept, eb::bigint nToUserId);

COM 接口：EB_VideoAck([in] ULONGLONG nCallId, [in] VARIANT_BOOL bAccept, [in] ULONGLONG nToUserId);

接受视频通话（加入视频会议）或拒绝视频通话请求；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **bool bAccept** | | 是否接收（加入）视频通讯 true/false |

| | | |
|---|---|---|
| **eb::bigint nToUserId** | 0 | 拒绝可接收指定用户 ID 的视频，用于视频会议 |

## 2.14.8. EB_VideoEnd

int EB_VideoEnd(eb::bigint nCallId);

COM 接口：EB_VideoEnd([in] ULONGLONG nCallId);

结束视频通话或退出视频会议；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nCallId** | | 会话编号 |

## 2.14.9. EB_GetUserVideoId

bool EB_GetUserVideoId(eb::bigint nCallId,std::vector<EB_UserVideoInfo>& pOutMemberVideoInfo,std::vector<EB_UserVideoInfo>& pOutOnVideoInfo,int& pOutMyVideoId) const;

COM 接口：EB_GetUserVideoId([in] ULONGLONG nCallId, [out] VARIANT* pOutMemberVideoInfo, [out] VARIANT* pOutOnVideoInfo, [out,retval] ULONG* pOutMyVideoId);

获取会话，所有用户视频编号；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nCallId** | | 会话编号 |
| **std::vector<EB_UserVideoInfo>& pOutMemberVideoId** | | [输出]成员用户视频编号 |
| **std::vector<EB_UserVideoInfo>& pOutOnVideoId** | | [输出]打开视频成员用户视频编号 |
| **int& pOutMyVideoId** | | [输出]我的视频编号 |

## 2.14.10. EB_OpenLocalVideo

int EB_OpenLocalVideo(eb::bigint nCallId,int nLocalVideoIndex,PAudioDataCallBack cbAudio,void* pParam,int& pOutVideoMediaType);

COM 接口：EB_OpenLocalVideo([in] ULONGLONG nCallId, [in] SHORT nLocalVideoIndex, [in] IDispatch* pAudioDataCallback, [in] ULONG nUserData, [out,retval] SHORT* pOutVideoMediaType);

打开会话本地视频设备 ,根据视频请求 ,打开视频设备( 或声卡设备、或二个同时打开 )；

返回本地用户视频编号， - 1 表示失败

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **int nLocalVideoIndex** | | 打开指定索引位置视频设备 |
| **PAudioDataCallBack cbAudio** | | 音频数据回调函数 |
| **void* pParam** | | 自定义回调参数 |
| **int& pOutVideoMediaType** | | [输出]视频媒体类型 |

## 2.14.11. EB_OpenVideoByUserId

bool EB_OpenVideoByUserId(eb::bigint nCallId,eb::bigint nUserId,PAudioDataCallBack cbAudio,void* pParam);

COM 接口：EB_OpenVideoByUserId([in] ULONGLONG nCallId, [in] ULONGLONG nUserId, [in] IDispatch* pAudioDataCallback, [in] ULONG nUserData);

打开会话远程用户视频；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **eb::bigint nUserId** | | 会话用户帐号 |
| **PAudioDataCallBack cbAudio** | | 音频数据回调函数 |
| **void* pParam** | | 自定义回调参数 |

## 2.14.12. EB_OpenVideoByUserVideoId

bool EB_OpenVideoByUserVideoId(eb::bigint nCallId,int nUserVideoId,PAudioDataCallBack cbAudio,void* pParam);

COM 接口 :EB_OpenVideoByUserVideoId([in] ULONGLONG nCallId, [in] ULONG nUserVideoId, [in] IDispatch* pAudioDataCallback, [in] ULONG nUserData);

打开会话远程用户视频；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nCallId** | | 会话编号 |
| **int nUserVideoId** | | 会话用户视频编号 |
| **PAudioDataCallBack cbAudio** | | 音频数据回调函数 |

| | |
|---|---|
| **void* pParam** | 自定义回调参数 |

### 2.14.13. EB_CloseAllVideo

void EB_CloseAllVideo(eb::bigint nCallId);

COM 接口：EB_CloseAllVideo([in] ULONGLONG nCallId);

关闭会话所有视频资源；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **eb::bigint nCallId** | | 会话编号 |

## 2.15. 协同功能函数

### 2.15.1. EB_GetSubscribeFuncList

int EB_GetSubscribeFuncList(EB_FUNC_LOCATION nFuncLocation,std::vector<EB_SubscribeFuncInfo>& pOutFuncInfo) const;

COM 接口：EB_GetSubscribeFuncList([in] ULONG nLocation, [out,retval] VARIANT* pVal);

获取客户端指定显示位置协同功能信息；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **EB_FUNC_LOCATION nFuncLocation** | | 指定要显示位置 |
| **std::vector<EB_SubscribeFuncInfo>& pOutFuncInfo** | | [输出]该显示位置的协同功能信息 |

### 2.15.2. EB_GetSubscribeFuncSize

int EB_GetSubscribeFuncSize(EB_FUNC_LOCATION nFuncLocation) const;

COM 接口：EB_GetSubscribeFuncSize([in] ULONG nLocation, [out,retval] ULONG* pVal);

获取客户端指定显示位置协同功能数量；

| 参数 | 默认值（非空） | 描述 |
|---|---|---|
| **EB_FUNC_LOCATION nFuncLocation** | | 指定要显示位置 |

### 2.15.3. EB_GetSubscribeFuncInfo

int EB_GetSubscribeFuncInfo(eb::bigint nSubId, EB_SubscribeFuncInfo* pOutFuncInfo) const;

COM 接口：EB_GetSubscribeFuncInfo([in] ULONGLONG nSubId, [out,retval] IDispatch* pVal);

获取客户端指定显示位置协同功能信息；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nSubId** | | 应用订购 ID |
| **EB_SubscribeFuncInfo* pOutFuncInfo** | | [输出]应用订购信息 |

## 2.15.4. EB_GetSubscribeFuncUrl

std::string EB_GetSubscribeFuncUrl(eb::bigint nSubscribeId,eb::bigint nCallId) const;

COM 接口：EB_GetSubscribeFuncUrl([in] ULONGLONG nSubscribeId, [in] ULONGLONG nCallId, [out,retval] BSTR* pVal);

获取协同功能业务 URL 地址；

| 参数 | 默认值（非空） | 描述 |
| --- | --- | --- |
| **eb::bigint nSubscribeId** | | 协同功能申请 ID |
| **eb::bigint nCallId** | 0 | 会话 ID |

# 3.附录

## 3.1. 协同功能显示位置

```
/*=============================================================
应用功能定位类型
=============================================================*/
typedef enum EB_FUNC_LOCATION
{
    EB_FUNC_LOCATION_MAINFRAME_BTN                  = 0x0001    // 主界面按钮
    , EB_FUNC_LOCATION_APPFRAME_BTN                 = 0x0002    // 应用面板按
钮
    , EB_FUNC_LOCATION_RIGHT_CLICK_MENU_MAINFRAME  = 0x0010    // 主界面右键菜单
    , EB_FUNC_LOCATION_RIGHT_CLICK_MENU_USER        = 0x0020    // 用户右键菜单
    , EB_FUNC_LOCATION_RIGHT_CLICK_MENU_GROUP       = 0x0040    // 群组（部门）右
键菜单
    , EB_FUNC_LOCATION_RIGHT_CLICK_MENU_ENTERPRISE  = 0x0080    // 企业右键菜单
    , EB_FUNC_LOCATION_GROUP_CHAT_BTN1              = 0x0100    // 群组会话上面按
钮
    , EB_FUNC_LOCATION_GROUP_CHAT_BTN2              = 0x0200    // 群组会话下面按
钮
    , EB_FUNC_LOCATION_USER_CHAT_BTN1               = 0x0400    // 用户会话上面按
钮
    , EB_FUNC_LOCATION_USER_CHAT_BTN2               = 0x0800    // 用户会话下面按
钮
    , EB_FUNC_LOCATION_CALL_CHAT =
EB_FUNC_LOCATION_GROUP_CHAT_BTN1|EB_FUNC_LOCATION_GROUP_CHAT_BTN2|EB_FUNC_
LOCATION_USER_CHAT_BTN1|EB_FUNC_LOCATION_USER_CHAT_BTN2

    };
```