LONG-D.COM

龙笛插件系统技术白皮书

C++版

TONY

2012/7/22

详细介绍龙笛插件系统的依赖文件、导出函数、各种服务接口、各种回调接口以及各种相关结构体、事件、枚举和常量。

目录

第一章 概述

随着企业信息化程度的越来越高，即时通讯软件（IM）在企业信息化进程中不断被强化，作为信息传递的中枢，与 OA、CRM、E-mail、SMS 等软件的联系越来越紧密，并朝着平台化方向演变。为了方便企业用户深度整合各种 OA 相关的软件，个性化地定制属于自己的企业即时通讯系统，从龙笛 1.5.0.25 版本开始，引入了龙笛插件体系。

在最初的版本中，龙笛插件体系仅支持客户端插件。随着龙笛版本的不断完善，插件体系也得到了发展。目前已支持三种不同的插件，分别是客户端插件、服务端插件、后台管理系统插件。三种插件从各个方面进一步完善了龙笛插件体系，为企业提供了更深入的二次开发和深度整合能力。

本文档着眼于全面介绍龙笛插件体系的细节，包括依赖文件、导出函数、各种服务接口、各种回调接口以及各种相关结构体、事件、枚举和常量。因此，本文档并不是一份有关龙笛插件开发的入门介绍。如果需要简要的入门开发指南，可以参见《龙笛插件开发快速入门手册》。

龙笛插件开发 SDK 随安装包一起发布，通常解压安装包后，有一个名为 SDK 的文件夹。该文件夹包括龙笛插件开发的全部头文件、库文件：

ZJIMDef.h
EIMPluginServiceDef.h
EIMPluginServiceInc.h
EIMPluginService.lib
EIMPLGPluginServiceDef.h
EIMPLGPluginServiceInc.h
EIMPlgPluginService.lib
EIMSysMgrPluginServiceDef.h
EIMSysMgrPluginServiceInc.h
EIMSysMgrPluginService.lib

目前的龙笛插件体系是以 C++ header/lib/dll 的方式工作的，由 C++ header 定义服务接口，lib 提供接口函数的二进制链接，dll 方式加载插件。因此，开发龙笛插件需要使用 C++语言进行。

第二章 客户端插件

依赖文件

- 头文件：EIMPluginServiceInc.h
- 库文件：EIMPluginService.lib

插件模块导出函数

一个客户端插件的入口 DLL 必须导出如下三个函数才能被系统加载并被识别为客户端插件：

- PluginInitialize
  插件被初始化时调用此函数，在此函数内插件通过 EIMPluginInfo 结构体设置插件 GUID、插件名称、描述等信息，并进行插件内部状态初始化、资源分配等任务。

  函数原型：
  BOOL PluginInitialize(EIMPluginInfo& stPluginInfo)

  函数参数：
  stPluginInfo　　EIMPluginInfo 结构体的引用，用于向系统返回插件信息。

  示例代码：

```
BOOL PluginInitialize(EIMPluginInfo& stPluginInfo)
{
    stPluginInfo.nSlotItems =
EIM_SLOT_CHATDLG_EDIT_TOOLBAR|EIM_SLOT_TRIBEDLG_EDIT_TOOLBAR;
    _tcscpy_s(stPluginInfo.szPluginGuid, 33,
_T("CA308C19099749ae9ABAA6C4BF24DF26"));
    _tcscpy_s(stPluginInfo.szPluginComp, 32, _T("龙笛开发组"));
    _tcscpy_s(stPluginInfo.szPluginDesc, 256, _T("本插件实现掷骰子、抽奖功能，试一下您的运气，在单聊和群中娱乐！"));
    _tcscpy_s(stPluginInfo.szPluginName, 32, EIM_PLUGIN_NAME);
    _tcscpy_s(stPluginInfo.szPluginUrl, 64,
_T("http://www.long-d.cn"));
```

```
    return TRUE;
}
```

- **PluginUninitialize**

插件被卸载时调用此函数。在此函数内可进行插件自身的资源卸载等任务。

函数原型：

```
BOOL PluginUninitialize(void)
```

- **PluginRegistered**

系统在插件被加载及初始化完成后调用此函数，用于向插件提供一次机会，以便插件可以向系统的各个插槽区(slot)注册插槽对象和注册事件处理，或执行其它依赖于插件路径或插件 ID 的事务。

函数原型：

```
BOOL PluginRegistered(LPCTSTR strCurrentPluginPath, UINT nPluginID)
```

函数参数：

strCurrentPluginPath	字符串型，当前插件的绝对路径
nPluginID	无符号整型，系统分配给当前插件的 ID

通常，插件需要以图标的形式出现在系统中某个界面的插槽区(slot)。比如，一些重要的插件常常在主界面的顶部工具栏上添加图标，或者是聊天窗口的工具栏上添加图标，甚至是在主界面的 TAB 标签区添加一个 TAB 标签，产生一个新的扩展区域。这样的功能，需要在此函数中调用系统提供的服务接口来添加插槽区(slot)对象，并添加必要的事件处理回调接口。

示例代码：

下面的代码将向聊天窗口和群窗口的工具栏注册一个插槽区对象，然后注册事件处理接口。插槽区对象通过 EIMSlotItemInfo 来定义，并通过 IEIMPluginManager 接口的 AddSlotItem 方法把插槽区对象注册到系统中。随后调用了 IEIMPluginManager 接口的 AddBaseProc 和 AddPluginNotifyProc 两个方法来添加两种事件处理的回调接口。

```
BOOL PluginRegistered(LPCTSTR strCurrentPluginPath, UINT nPluginID)
{
    EIMSlotItemInfo slot_item;
    slot_item.nSlotItemInnerID = (WORD)ID_TOOLBARMENU_CAST;
    slot_item.nSlotItemMenu = IDR_MENU;
```

```
    slot_item.nSlotItemType =
EIM_SLOT_CHATDLG_EDIT_TOOLBAR|EIM_SLOT_TRIBEDLG_EDIT_TOOLBAR;
    _tcscpy_s(slot_item.szSlotItemIcon, 32,
_T("lucky_icon.png"));
    _tcscpy_s(slot_item.szSlotItemText, 32, _T("骰子"));
    _tcscpy_s(slot_item.szSlotItemToolTip, 32, _T("掷骰子、抽奖
"));

    IEIMPluginManager* pPluginManager = EIMGetPluginManager();
    if (pPluginManager != NULL)
    {
        pPluginManager->AddSlotItem(strCurrentPluginPath,
nPluginID, slot_item);

        pPluginManager->AddBaseProc(nPluginID, &procNotify);
        pPluginManager->AddPluginNotifyProc(nPluginID,
EIM_PLUGIN_ID, &procNotify);
    }

    return TRUE;
}
```

插件管理器接口

插件管理器接口 IEIMPluginManager 由系统提供，插件调用此接口主要用于添加、移除插槽区对象及各种事件处理的回调接口。

插件调用插件管理器接口 IEIMPluginManager 前，必须先获得该接口的指针，通过调用全局函数 EIMGetPluginManager 来实现。

该函数原型为：

```
__declspec(dllexport) IEIMPluginManager* EIMGetPluginManager();
```

插件管理器接口 IEIMPluginManager 主要提供了如下的方法供插件调用：

- UINT AddSlotItem(LPCTSTR strCurrentPluginPath, UINT nPluginID, const EIMSlotItemInfo& item)

  功能：添加插槽区对象。

  返回：成功返回插槽区对象ID，失败返回0。

  参数：

strCurrentPluginPath	字符串型，当前插件绝对路径
nPluginID	无符号整型，插件ID
item	EIMSlotItemInfo结构体，描述插槽区对象

- BOOL RemoveSlotItem(UINT nPluginID, UINT nItemID)

功能：移除插槽区对象。
返回：成功返回TRUE，失败返回FALSE。
参数：
nPluginID　　无符号整型，插件ID
nItemID　　无符号整型，插槽区对象ID

- BOOL AddBaseProc(UINT nPluginID, IEIMPluginBaseProc* pBaseProc)
功能：添加基本消息处理回调接口
返回：成功返回TRUE，失败返回FALSE。
参数：
nPluginID　　无符号整型，插件ID
pBaseProc　　IEIMPluginBaseProc接口的指针，该接口由插件实现，用于处理基本的消息

- void RemoveBaseProc(UINT nPluginID, IEIMPluginBaseProc* pBaseProc)
功能：添加
返回：无
参数：
nPluginID　　无符号整型，插件ID
pBaseProc　　IEIMPluginBaseProc接口的指针，该接口由插件实现，用于处理基本的消息

- BOOL AddMessageProc(UINT nPluginID, IEIMPluginMessageProc* pMessageProc)
功能：添加即时通讯消息处理回调接口
返回：成功返回TRUE，失败返回FALSE。
参数：
nPluginID　　无符号整型，插件ID
pMessageProc　IEIMPluginMessageProc接口的指针，该接口由插件实现，用于处理即时通讯消息

- void RemoveMessageProc(UINT nPluginID, IEIMPluginMessageProc* pMessageProc)
功能：添加
返回：无
参数：
nPluginID　　无符号整型，插件ID
pMessageProc　IEIMPluginMessageProc接口的指针，该接口由插件实现，用于处理即时通讯消息

- BOOL AddPluginNotifyProc(UINT nPluginID, LPCTSTR lpPluginID,

IEIMPluginNotifyProc* pNotifyProc)
功能：添加插件通知处理回调接口
返回：成功返回TRUE，失败返回FALSE。
参数：
nPluginID　　无符号整型，插件ID
lpPluginID　　字符串型，插件字符串型ID，建议此ID由以圆点分隔的多个字符串组成，包含公司名称和插件名称，以保证插件ID的唯一性
pNotifyProc　IEIMPluginNotifyProc接口指针，该接口由插件实现，用于处理插件的通知消息。

- void RemovePluginNotifyProc(LPCTSTR lpPluginID)
功能：移除插件通知处理回调接口
返回：无
参数：
lpPluginID　　字符串型，插件字符串型ID

插件服务接口

为了让插件更好地控制系统，调用系统的功能，如获取当前登录用户的 ID 和状态，调用系统消息发送功能等，插件系统提供了多个服务接口供插件使用。

在使用这些服务接口前，插件需要先调用相应的全局函数获得这些接口的实例指针。

例如：

```
IEIMBaseService* pBaseService = EIMGetBaseService();
if (pBaseService != NULL)
{
    TCHAR szUserName[100];
    szUserName[0] = 0;
    pBaseService->GetContactName(strSenderID, szUserName, 100);
}
```

- **基础服务接口：IEIMBaseService**
该服务接口为插件提供一些最基础的服务，例如：获取龙笛系统的版本号、登录用户的 ID、名称。

  - void GetAppVersion(int& major, int& minor, int& modify, int& build)
  功能：获取龙笛系统的版本号
  返回：无
  参数：
  major　　整型，主版本号

minor　　整型，次版本号
modify　　整型，修订号
build　　整型，编译次序号

- void GetCompanyName(CString& strCompanyName)
  功能：获取当前已注册的公司名称
  返回：无
  参数：
  strCompanyName　　字符串型，公司名称

- const CString& GetCurrentUserID()
  功能：获取当前用户的ID
  返回：字符串型，用户ID
  参数：无

- long GetCurrentUserStatus()
  功能：获取当前用户的在线状态
  返回：长整型，表示在线状态。在线状态的定义请参见IMUserStatus的定义。
  参数：无

- int GetCurrentUserName(TCHAR szName[], int nBufferLen)
  功能：获取当前用户姓名
  返回：当前用户姓名的字符串长度
  参数：
  szName　　字符数组
  nBufferLen　　字符数组的长度

- BOOL SetCurrentUserStatus(long lStatus)
  功能：设置当前用户的状态
  返回：成功返回TRUE，失败返回FALSE。
  参数：
  lStatus　　长整型，表示在线状态。在线状态的定义请参见IMUserStatus 的定义。

- BOOL SetCurrentUserName(const TCHAR* szName)
  功能：更改当前用户的姓名
  返回：成功返回TRUE，失败返回FALSE。
  参数：
  szName　　字符串指针，指向新的姓名字符串

- void Logoff()
  功能：注销当前已登录的用户
  返回：无

参数：无

- void Exit()
  功能：退出龙笛系统
  返回：无
  参数：无

- int GetTribeMemberCount(const CString& strTribeID)
  功能：获取指定的群包含的成员数目
  返回：整型，成员数目
  参数：
  strTribeID　字符串型，指定群的ID

- int GetTribeMemberIDs(const CString& strTribeID, LPTSTR* ppMemberIDs)
  功能：获取群成员列表
  返回：群成员列表的数目
  参数：
  strTribeID　字符串型，指定群的ID
  ppMemberIDs　字符串指针的指针，用于返回成员列表的字符串型ID,指向的内存由本函数分配，在调用完毕由调用者释放。

- int GetCurrentUserPath(TCHAR szBuffer[], int nBufferLen)
  功能：获取当前用户的绝对路径
  返回：整型，用户路径字符串的长度
  参数：
  szBuffer　字符数组
  nBufferLen　字符数组的长度

- int GetContactName(const CString& strContactID, TCHAR szBuffer[], int nBufferLen)
  功能：获取指定联系人的姓名
  返回：整型，联系人姓名字符串的长度
  参数：
  strContactID　字符串型，联系人的ID
  szBuffer　字符数组
  nBufferLen　字符数组的长度

- int GetContactIconFile(const CString& strContactID, BOOL bOnline, TCHAR szBuffer[], int nBufferLen)
  功能：获取联系人ICON文件的本地绝对路径
  返回：整型，ICON文件字符串的长度

参数：

strContactID	整整，联系人ID
bOnline	布尔，指定获取在线图标，还是离线图标
szBuffer	字符数组
nBufferLen	字符数组的长度

- int GetContacts(std::vector<ContactBriefInfo*>& vecContacts)

  功能：获取所有联系人

  返回：整型，联系人数目

  参数：

  vecContacts vector，用于返回所有联系人，联系人结构体参见 ContactBriefInfo 的定义

- void ReleaseContacts(std::vector<ContactBriefInfo*>& vecContacts)

  功能：释放所获得的联系人列表

  返回：无

  参数：

  vecContacts vector，联系人列表，联系人结构体参见 ContactBriefInfo 的定义

- void GetMainServiceIPPort(DWORD& dwServerIpAddr, unsigned short& usServerPort)

  功能：获取主服务器的IP地址和端口

  返回：无

  参数：

dwServerIpAddr	DWORD，服务器IP地址
usServerPort	short，服务器端口

- void GetFileServiceIPPort(DWORD& dwServerIpAddr, unsigned short& usServerPort)

  功能：获取文件服务器的IP地址和端口

  返回：无

  参数：

dwServerIpAddr	DWORD，服务器IP地址
usServerPort	short，服务器端口

- void GetDataServiceIPPort(DWORD& dwServerIpAddr, unsigned short& usServerPort)

  功能：获取数据服务器的IP地址和端口

  返回：无

  参数：

dwServerIpAddr	DWORD，服务器IP地址

usServerPort　　short，服务器端口

- void GetPluginServiceIPPort(DWORD& dwServerIpAddr, unsigned short& usServerPort)
  功能：获取插件服务器的IP地址和端口
  返回：无
  参数：
  dwServerIpAddr　DWORD，服务器IP地址
  usServerPort　　short，服务器端口

- void SetMutexWnd(HWND hWnd)
  功能：设置指定的窗口为互斥窗口。设为互斥窗口后，先前的互斥窗口将被关闭。系统中同一时间只会存在一个互斥窗口。
  返回：无
  参数：
  hWnd　　HWND，窗口句柄

- HWND GetMutexWnd()
  功能：获取当前的互斥窗口
  返回：HWND，窗口句柄
  参数：无

- void AddAutoDeleteWnd(HWND hWnd)
  功能：添加指定的窗口到自动删除窗口队列中。该窗口关闭后，将自动删除。
  返回：无
  参数：
  hWnd　　HWND，窗口句柄

- void RemoveAutoDeleteWnd(HWND hWnd)
  功能：从自动删除窗口队列中移出指定的窗口。
  返回：无
  参数：
  hWnd　　HWND，窗口句柄

- HWND FindUniqueWnd(LPCTSTR strWndName)
  功能：查找指定名称的唯一实例窗口。
  返回：HWND，窗口句柄
  参数：
  strWndName　　字符串型，待查找的唯一实例窗口名称

- BOOL InsertUniqueWnd(LPCTSTR strWndName, HWND hWnd)
  功能：添加指定的窗口到唯一实例窗口列表中。该窗口被添加后，同一时间将只能存在唯一的一个实例。

返回：成功返回TRUE，失败返回FALSE。
参数：
strWndName 字符串型，窗口名称，窗口名称由使用方自定义，为保证其唯一性，需要包含特定的字符串和一定的长度
hWnd HWND，窗口句柄

- BOOL ReleaseUniqueWnd(LPCTSTR strWndName)
功能：释放唯一实例窗口。该窗口将被关闭和释放。
返回：成功返回TRUE，失败返回FALSE。
参数：
strWndName 字符串型，窗口名称

- BOOL RemoveUniqueWnd(LPCTSTR strWndName)
功能：移除唯一实例窗口。该窗口仅从唯一实例列表中移除，不会被关闭和释放。
返回：成功返回TRUE，失败返回FALSE。
参数：
strWndName 字符串型，窗口名称

- void GetAttachedWndPos(const CSize& sizeWnd, CPoint& point)
功能：获取附着到主窗口的合适的窗口位置
返回：无
参数：
sizeWnd CSize，传入窗口的尺寸
point CPoint，传出窗口的位置

- void AdjustWindowPos(CWnd* pWnd, CWnd* pParentWnd = NULL, PluginChildWndPosition emPos = Plugin_Child_Attach, BOOL bSameHeight = FALSE, BOOL bLeftPriority = FALSE)
功能：自动调整窗口的位置和大小，使之附着到指定的父窗口的合适位置
返回：无
参数：
pWnd CWnd，需要被调整的窗口指针
pParentWnd CWnd，父窗口的指针
emPos PluginChildWndPosition枚举值，指定附着方式
bSameHeight BOOL，是否与父窗口高度相同
bLeftPriority BOOL，是否左边附着优先考虑

- void SetMainFrmMainToolBarItemText(UINT nPluginID, UINT nItemID, LPCTSTR strItemText)
功能：设置主界面主工具栏的插件图标的文字
返回：无
参数：

nPluginID　　UINT，插件ID
nItemID　　UINT，工具栏图标ID
strItemText　　字符串型，图标项的文字

- void ShowSystemNotify(LPCTSTR szTitle, LPCTSTR szContent, LPCTSTR szUrl, UINT nPluginID, UINT nParam)

功能：显示一条系统通知
返回：无
参数：
nPluginID　　UINT，插件ID
szTitle　　字符串型，系统通知的标题
szContent　　字符串型，系统通知的内容
szUrl　　字符串型，系统通知的URL
nParam　　UINT，额外的自定义参数

- INT_PTR ShowSelectContactDlg(HWND hWndParent, LPCTSTR strWndTitle, DWORD dwMask, LPCTSTR strDefContacts, LPCTSTR strDefTribes, LPTSTR* pstrContacts, LPTSTR* pstrTribes)

功能：打开联系人选择器窗口
返回：INT_PTR，窗口关闭的返回值，如ID_OK\ID_CANCEL等
参数：
hWndParent　　HWND，指定选择器窗口的父窗口
strWndTitle　　字符串型，窗口标题
dwMask　　DWORD，掩码，用于指定显示何种选择器，目前其取值为 SEL_CONTACT_MASK_CONTACT、SEL_CONTACT_MASK_TRIBE
strDefContacts　　字符串型，指定初始的联系人ID列表，多个联系人ID之间以英文逗号分隔
strDefTribes　　字符串型，指定初始的群ID列表，多个群 ID之间以英文逗号分隔
pstrContacts　　字符串指针的指针，返回选择的联系人ID，多个联系人ID之间以英文逗号分隔。内存分配在此函数中进行，调用方需要释放。
pstrTribes　　字符串指针的指针，返回选择的群ID，多个群ID之间以英文逗号分隔。内存分配在此函数中进行，调用方需要释放。

- BOOL SaveCurrentUserOption(LPCTSTR strSection, LPCTSTR strKey, LPCTSTR strValue)

功能：保存当前用户的选项
返回：成功返回TRUE，失败返回FALSE。
参数：
strSection　　字符串型，指定选项的section
strKey　　字符串型，指定选项的key
strValue　　字符串型，指定选项的value

- BOOL LoadCurrentUserOption(LPCTSTR strSection, LPCTSTR strKey, TCHAR szBuffer[], int nBufferLen)

  功能：加载当前用户的选项
  返回：成功返回TRUE，失败返回FALSE。
  参数：

strSection	字符串型	指定选项的section
strKey	字符串型	指定选项的key
szBuffer	字符数组	返回指定选项的value
nBufferLen	字符数组的长度	

■ **消息服务接口：IEIMMessageService**

该服务接口为插件提供即时消息相关的服务，例如：打开聊天窗口、发送即时消息、发送插件通知消息。

- BOOL OpenChatDlg(const CString& strContactID, BOOL bActive)

  功能：打开与指定联系人的聊天窗口
  返回：成功返回TRUE，失败返回FALSE。
  参数：

strContactID	字符串型	联系人ID
bActive	布尔型	窗口是否被激活

- void CloseChatDlg(const CString& strContactID)

  功能：关闭与指定联系人的聊天窗口
  返回：无
  参数：

strContactID	字符串型	联系人ID

- BOOL OpenTribeDlg(const CString& strTribeID, BOOL bActive)

  功能：打开指定的群聊天窗口
  返回：成功返回TRUE，失败返回FALSE。
  参数：

strTribeID	字符串型	群ID
bActive	布尔型	窗口是否被激活

- void CloseTribeDlg(const CString& strTribeID)

  功能：关闭群聊天窗口
  返回：无
  参数：

strTribeID	字符串型	群ID

- BOOL OpenGroupSendDlg(const CString& strContactIDs, const TCHAR* szMsg)

  功能：打开群发消息窗口

返回：成功返回TRUE，失败返回FALSE。
参数：
strContactIDs 字符串型，消息接收者ID，多个ID之间以英文逗号分隔
szMsg 字符串型指针，群发的消息内容

- BOOL SendMsg(const CString& strContactID, const TCHAR* szMsg)
功能：发送即时消息
返回：成功返回TRUE，失败返回FALSE。
参数：
strContactID 字符串型，消息接收者ID
szMsg 字符串型指针，消息内容

- BOOL InputMsg(const CString& strContactID, const TCHAR* szMsg)
功能：输入消息到聊天窗口
返回：成功返回TRUE，失败返回FALSE。
参数：
strContactID 字符串型，消息接收者ID
szMsg 字符串型指针，消息内容
备注：该方法暂时实现

- BOOL SendTribeMsg(const CString& strTribeID, const TCHAR* szMsg)
功能：发送群消息
返回：成功返回TRUE，失败返回FALSE。
参数：
strTribeID 字符串型，群ID
szMsg 字符串型指针，消息内容

- BOOL InputTribeMsg(const CString& strTribeID, const TCHAR* szMsg)
功能：
返回：成功返回TRUE，失败返回FALSE。
参数：
strTribeID 字符串型，群ID
szMsg 字符串型指针，消息内容
备注：该方法暂时实现

- BOOL SendGroupMsg(LPCTSTR lpContactIDs, const TCHAR* szMsg)
功能：发送群发消息
返回：成功返回TRUE，失败返回FALSE。
参数：
lpContactIDs 字符串型指针，消息接收者ID，多个ID之间以英

文逗号分隔
szMsg 字符串型指针，群发的消息内容

- BOOL SendAutoReplayMsg(const CString& strContactID, const TCHAR* szMsg)
功能：发送自动回复消息
返回：成功返回TRUE，失败返回FALSE。
参数：
strContactID 字符串型，消息接收者ID
szMsg 字符串型指针，消息内容

- BOOL OpenSmsDlg(const TCHAR* szMobileNum)
功能：打开短信发送窗口
返回：成功返回TRUE，失败返回FALSE。
参数：
szMobileNum 字符串型，移动电话号码

- BOOL SendSms(const TCHAR* szMobileNum, const TCHAR* szSms)
功能：发送短信
返回：成功返回TRUE，失败返回FALSE。
参数：
szMobileNum 字符串型，移动电话号码
szSms 字符串型，短信内容

- BOOL GroupSendSms(TCHAR szMobileNums[][16], const TCHAR* szSms)
功能：群发短信
返回：成功返回TRUE，失败返回FALSE。
参数：
szMobileNums 字符串型数组，移动电话号码列表
szSms 字符串型，短信内容

- BOOL InsertSysMsg2ChatDlg(const CString& strContactID, LPCTSTR lpSender, LPCTSTR lpMsg)
功能：向聊天窗口插入系统消息
返回：成功返回TRUE，失败返回FALSE。
参数：
strContactID 字符串型，消息接收者ID
lpSender 字符串型指针，发送者名称
lpMsg 字符串型指针，消息内容

- BOOL InsertSysMsg2TribeDlg(const CString& strTribeID, LPCTSTR lpSender, LPCTSTR lpMsg)
功能：向群聊天窗口插入系统消息

返回：成功返回TRUE，失败返回FALSE。
参数：
strTribeID　字符串型，群ID
lpSender　字符串型指针，发送者名称
lpMsg　字符串型指针，消息内容

- BOOL SendPluginNormalNotify(const CString& strContactID, LPCTSTR lpPluginID, LPCTSTR lpPluginName, LPCTSTR lpMsg)
功能：发送插件的普通通知消息
返回：成功返回TRUE，失败返回FALSE。
参数：
strContactID　字符串型，接收者的联系人ID
lpPluginID　字符串型指针，插件ID
lpPluginName　字符串型指针，插件名称
lpMsg　字符串型指针，消息内容

- BOOL SendPluginTribeNotify(const CString& strTribeID, LPCTSTR lpPluginID, LPCTSTR lpPluginName, LPCTSTR lpMsg)
功能：发送插件的群通知消息
返回：成功返回TRUE，失败返回FALSE。
参数：
strContactID　字符串型，接收者的联系人ID
lpPluginID　字符串型指针，插件ID
lpPluginName　字符串型指针，插件名称
lpMsg　字符串型指针，消息内容

- BOOL SendPluginGroupNotify(LPCTSTR lpContactIDs, LPCTSTR lpPluginID, LPCTSTR lpPluginName, LPCTSTR lpMsg)
功能：发送插件群发通知消息
返回：成功返回TRUE，失败返回FALSE。
参数：
lpContactIDs　字符串型指针，接收者的联系人ID，多个ID之间以英文逗号分隔
lpPluginID　字符串型指针，插件ID
lpPluginName　字符串型指针，插件名称
lpMsg　字符串型指针，消息内容

插件网络通信服务

当插件需要网络通信服务时，不必自行从基础的功能开始开发，可以使用系统提供的基本网络通信服务。基本的网络通信服务基于套接字（socket）进行封装，为插件提供连接服务器及双向通信的能力。

插件网络通信服务通过 IEIMPluginClientSocket 接口提供，插件须调用全局函数 EIMCreatePluginClientSocket 创建一个指向 IEIMPluginClientSocket 接口的对象实例。然后通过调用 IEIMPluginClientSocket 接口的方法进行通信。

通信完毕不再使用时，插件须调用全局函数 EIMReleasePluginClientSocket 释放先前获得的 IEIMPluginClientSocket 接口指针。

IEIMPluginClientSocket 接口提供了如下的方法：

- void SetTimeOut(DWORD dwConnectTimeOut, DWORD dwSendTimeOut, DWORD dwRecvTimeOut)
  功能：设置连接、发送和接收的超时
  返回：无
  参数：
  dwConnectTimeOut DWORD，连接超时设置，毫秒
  dwSendTimeOut DWORD，发送超时设置，毫秒
  dwRecvTimeOut DWORD，接收超时设置，毫秒

- BOOL Connect(const CString& strIpAddr, unsigned short usPort)
  功能：连接服务器
  返回：成功返回TRUE，失败返回FALSE
  参数：
  strIpAddr 字符串型，服务器IP地址
  usPort 无符号短整型，服务器端口

- BOOL Connect(DWORD dwIpAddr, unsigned short usPort)
  功能：连接服务器
  返回：成功返回TRUE，失败返回FALSE
  参数：
  dwIpAddr DWORD，服务器IP地址（主机顺序）
  usPort 无符号短整型，服务器端口

- BOOL ConnectByHostName(const CString& strServerName, unsigned short usPort)
  功能：连接服务器
  返回：成功返回TRUE，失败返回FALSE
  参数：
  strServerName 字符串型，主机名称
  usPort 无符号短整型，服务器端口

- DWORD GetHostIpAddr() const
  功能：获取本机的IP地址
  返回：DWORD，IP地址，主机顺序

参数：无

- void Disconnect()
  功能：断开与服务器的连接
  返回：无
  参数：无

- BOOL Send(LPBYTE pData, int nDataSize)
  功能：发送消息
  返回：成功返回TRUE，失败返回FALSE
  参数：
  pData　　BYTE指针，指向消息内容
  nDataSize　int，消息内容的字节长度

- BOOL RecvMsg(BYTE** ppData, int& nDataSize)
  功能：接收消息
  返回：成功返回TRUE，失败返回FALSE
  参数：
  ppData　　BYTE型指针的指针，其内存由本方法分配，调用方须调用ReleaseMsg释放该指针
  nDataSize　int，返回接收到的消息长度

- void ReleaseMsg(BYTE** ppData)
  功能：释放通过RecvMsg分配的内存
  返回：无
  参数：
  ppData　　BYTE型指针的指针，由RecvMsg返回的双指针

- long GetLastError()
  功能：获取最后一次错误的ID
  返回：错误ID
  参数：无

插件回调接口

当插件需要处理系统的消息，或插件自己的通知时，插件需要实现相应的回调接口。回调接口由插件按规范实现，系统在事件发生时进行调用。

目前有三类回调接口，插件根据需要有选择性地实现：基本消息处理接口、即时消息处理接口、插件内部通知事件处理接口。

每种回调接口实现后，插件需要把接口注册到插件系统中。注册上述三种回调接口分别是由 IEIMPluginManager 接口的 AddBaseProc、AddMessageProc 和

AddPluginNotifyProc 实现的。

- **基本消息处理接口：IEIMPluginBaseProc**
  - BOOL OnWindowsMessage(UINT nMsg, WPARAM wParam, LPARAM lParam, const CString& strIDContactOrTribe, BOOL bIsTribe)
    功能：该方法用于处理一般性的 windows 消息，诸如按钮点击、菜单选中之类。插件注册到插槽区的对象被点击，菜单被用户选中，是通过本方法通知给插件的。
    返回：如果消息已被处理，则返回 TRUE；否则，对于不感兴趣的消息，返回 FALSE
    参数：
    nMsg、wParam、lParam 与 Windows 的消息处理函数意义相同
    strIDContactOrTribe 字符串型，指示联系人或群的 ID
    bIsTribe 布尔型，指示 strIDContactOrTribe 参数是否是群 ID

  - BOOL OnCurrentUserStatusChanged(IMUserStatus emUserStatus)
    功能：指示当前用户状态改变
    返回：已响应返回TRUE，否则为FALSE
    参数：
    emUserStatus IMUserStatus枚举，指示新的状态。参见IMUserStatus定义。

  - BOOL OnCurrentUserLogon(IMUserStatus emUserStatus)
    功能：指示用户登录成功
    返回：已响应返回TRUE，否则为FALSE
    参数：
    emUserStatus IMUserStatus枚举，指示当前状态。参见IMUserStatus定义。

  - BOOL OnCurrentUserLogoff()
    功能：指示当前用户注销
    返回：已响应返回TRUE，否则为FALSE
    参数：无

  - BOOL OnContactStatusChanged(const CString& strContactID, IMUserStatus emUserStatus)
    功能：指示联系人状态改变
    返回：已响应返回TRUE，否则为FALSE
    参数：
    strContactID 字符串型，联系人ID
    emUserStatus IMUserStatus枚举，指示新的状态。参见IMUserStatus定义。

- BOOL OnAppExit()
  功能：指示龙笛系统即将退出
  返回：已响应返回TRUE，否则为FALSE
  参数：无

■ **即时消息处理接口：IEIMPluginMessageProc**
注意：本接口的回调暂未实现，将来的版本会实现，但有可能发生改变。

- BOOL OnNewMsg(const CString& strContactID, long lMsgType, TCHAR* szMsg)
- BOOL OnNewTribeMsg(const CString& strTribeID, TCHAR* szMsg)
- BOOL OnSendMsg(const CString& strContactID, long lMsgType, TCHAR* szMsg)
- BOOL OnSendTribeMsg(const CString& strTribeID, TCHAR* szMsg)
- BOOL OnNewSms(const TCHAR* szMobileNum, TCHAR* szSms)
- BOOL OnSendSms(const TCHAR* szMobileNum, TCHAR* szSms)
- BOOL OnNewSysMsg(const DATE* dtMsgTime, const TCHAR* szTitle, const TCHAR* szContent)
- BOOL OnNewEmailNotify(const DATE* dtMsgTime, const TCHAR* szSender, const TCHAR* szSubject, const TCHAR* szContent, UINT nNewMailCount)

■ **插件内部通知事件处理接口：IEIMPluginNotifyProc**

本接口有 2 个方法，一个用于接收普通即时消息，另一个用于接收群消息。

通常，插件自己内部的简单通信可通过调用 IEIMMessageService 接口的SendPluginNormalNotify方法和本接口的OnRecvNormalMsg方法来完成。相应的群内的插件通信可通过调用 SendPluginTribeNotify 和 OnRecvTribeMsg 来完成。

- BOOL OnRecvNormalMsg(const DATE& dtTime, const CString& strSenderID, LPCTSTR lpMsg, BOOL bOfflineMsg)
  功能：接收插件内部普通即时消息
  返回：通过返回值指示本消息是否被处理，若被处理则返回TRUE，否则，返回FALSE
  参数：

dtTime	DATE，消息发送时间
strSenderID	字符串型，发送者ID
lpMsg	字符串型，消息内容
bOfflineMsg	布尔值，是否为离线消息

- BOOL OnRecvTribeMsg(const DATE& dtTime, const CString& strSenderID, const CString& strTribeID, LPCTSTR lpMsg, BOOL bOfflineMsg)
  功能：接收插件内部群消息
  返回：通过返回值指示本消息是否被处理，若被处理则返回TRUE，否则，返回FALSE
  参数：

dtTime	DATE，消息发送时间
strSenderID	字符串型，发送者ID
strTribeID	字符串型，群ID
lpMsg	字符串型，消息内容
bOfflineMsg	布尔值，是否为离线消息

全局函数

- **EIMGetPluginManager**
  获取插件管理器对象的指针

  函数原型：
  ```
  __declspec(dllexport) IEIMPluginManager* EIMGetPluginManager();
  ```

- **EIMGetBaseService**
  获取基础服务接口的指针

  函数原型：
  ```
  __declspec(dllexport) IEIMBaseService* EIMGetBaseService();
  ```

- **EIMGetMessageService**
  获取消息服务接口的指针

  函数原型：
  ```
  __declspec(dllexport) IEIMMessageService*
  EIMGetMessageService();
  ```

- **EIMCreatePluginClientSocket**
  创建可供插件使用的客户端套接字（socket）

  函数原型：
  ```
  __declspec(dllexport) BOOL EIMCreatePluginClientSocket(LPCTSTR
  szPluginGUID, IEIMPluginClientSocket** ppClientSocket);
  ```

- **EIMReleasePluginClientSocket**

释放可供插件使用的客户端套接字（socket）

函数原型：

```
__declspec(dllexport) void
EIMReleasePluginClientSocket(IEIMPluginClientSocket**
ppClientSocket);
```

结构体及其它

- **EIMPluginInfo**

该结构体用于 PluginInitialize 函数调用时把插件信息传给系统。插件信息包含插件 GUID、插件名称等信息。

```
typedef struct tagEIMPluginInfo
{
    TCHAR          szPluginGuid[33];
    TCHAR          szPluginName[32];
    TCHAR          szPluginDesc[256];
    TCHAR          szPluginComp[32];
    TCHAR          szPluginUrl[64];
    UINT           nSlotItems;
}EIMPluginInfo, *LPEIMPluginInfo;
```

- szPluginGuid　　字符串型，插件 GUID
- szPluginName　　字符串型，插件名称
- szPluginDesc　　字符串型，插件描述
- szPluginComp　　字符串型，插件开发公司名称
- szPluginUrl　　字符串型，插件开发公司网址
- nSlotItems　　UINT，插件将在哪些区域添加插槽区对象

- **EIMSlotItemInfo**

该结构体用于向插件系统注册插槽区对象。注册插槽区对象一般至少需要提供插槽区对象类型，内部 ID、图标或文字等信息。

```
typedef struct tagEIMSlotItemInfo
{
    SHORT          nSlotItemInnerID;
    UINT           nSlotItemType;
    TCHAR          szSlotItemIcon[32];
    TCHAR          szSlotItemText[32];
    TCHAR          szSlotItemToolTip[64];
    UINT           nSlotItemMenu;
```

```
    UINT          nSlotItemWnd;
}EIMSlotItemInfo, *LPEIMSlotItemInfo;
```

- nSlotItemInnerID　　短整型，指定一个内部 ID
- nSlotItemType　　整型，插件类型的组合
- szSlotItemIcon　　字符串型，图标文件相对路径
- szSlotItemText　　字符串型，文字
- szSlotItemToolTip　　字符串型，Tooltip
- nSlotItemMenu　　整型，菜单资源 ID
- nSlotItemWnd　　暂未使用

■ **EIMTABSELCHANGED**

该结构体用于 WM_MAINTAB_SELECTED 消息处理时，把 LPARAM 参数造型为 EIMTABSELCHANGED，通过本结构体获得容器窗口的信息。

```
typedef struct tagEIMTABSELCHANGED
{
    HWND   hContainerWnd;
    RECT   rtContainerClient;
}EIMTABSELCHANGED, *LPEIMTABSELCHANGED;
```

- hContainerWnd　　HWND，容器窗口句柄
- rtContainerClient　　RECT，容器窗口位置和尺寸

■ **PluginChildWndPosition**

本枚举定义子窗口与父窗口的相对关系，目前只有两种：居中和附着。

```
enum PluginChildWndPosition
{
    Plugin_Child_Center,
    Plugin_Child_Attach,
};
```

■ **插件类型**

- EIM_SLOT_MAINFRM_TOP_TOOLBAR：主界面顶部工具栏
- EIM_SLOT_MAINFRM_BOTTOM_TOOLBAR：主界面底部工具栏
- EIM_SLOT_CHATDLG_TOP_TOOLBAR：聊天窗口顶部工具栏
- EIM_SLOT_CHATDLG_EDIT_TOOLBAR：聊天窗口编辑工具栏
- EIM_SLOT_TRIBEDLG_TOP_TOOLBAR：群窗口顶部工具栏
- EIM_SLOT_TRIBEDLG_EDIT_TOOLBAR：群窗口编辑工具栏
- EIM_SLOT_MAINFRM_TAB：主界面TAB标签区
- EIM_SLOT_NAMECARD_TOOLBAR：名片区工具栏

■ **消息**

消息（message）	WPARAM wParam	LPARAM lParam
WM_SYSNOTIFY_CLICKED	调用 ShowSystemNotify 的 nParam	unused
WM_MAINTAB_SELECTED	插件插槽对象（Slot Item）的 ID	EIMTABSELCHANGED 结构体指针，包括容器窗口句柄和窗口矩形尺寸
WM_MAINTAB_UNSELECTED	同上	同上
WM_MAINTAB_SIZE	CRect 对象指针，指示当前容器的尺寸	unused

第三章 服务端插件

依赖文件

- 头文件：EIMPLGPluginServiceInc.h
- 库文件：EIMPLGPluginService.lib

插件模块导出函数

一个服务端插件的入口 DLL 必须导出如下三个函数才能被系统加载并被识别为服务端插件：

- PLGPluginInitialize
  插件被初始化时调用的函数，在此函数内插件通过 EIMPLGPluginInfo 结构体设置插件 GUID、插件名称、描述等信息，并进行插件内部状态初始化、资源分配等任务。

  函数原型：
  BOOL PLGPluginInitialize(EIMPLGPluginInfo& stPLGPluginInfo)

- PLGPluginUninitialize
  插件被卸载时调用此函数，在此函数内可进行插件自身的资源卸载等任务。

  函数原型：
  BOOL PLGPluginUninitialize()

- PLGPluginNetMsg
  服务端接收到客户端插件发送的消息时，将根据插件的 ID 进行消息派发，插件通过此函数处理客户端插件发送的消息。

  函数原型：
  BOOL PLGPluginNetMsg(void* pSocketHandle, LPBYTE pMsgData, int nMsgDataLen)

  参数：

pSocketHandle　　void*，表示 Socket 的内部句柄，在调用 IEIMPlgSvrBaseService 接口的 SendToClient 发送消息时需要此参数
pMsgData　　字节数组，保存着消息的内容
nMsgDataLen　　整型，消息内容的长度

服务端服务接口

目前服务端向插件提供了基础的服务接口 IEIMPlgSvrBaseService，使用该接口可以与客户端进行通信。

插件通过全局函数 EIMGetPlgSvrBaseService 获取 IEIMPlgSvrBaseService 接口指针，然后调用该接口的方法。

IEIMPlgSvrBaseService 提供的方法：

- BOOL SendToClient(void* pSocketHandle, int nMsgType, LPBYTE pMsgData, int nMsgDataLen)
  功能：该方法向客户端插件发送一条消息
  返回：发送成功返回 TRUE，失败返回 FALSE
  参数：
  pSocketHandle　　void*，由 PLGPluginNetMsg 函数传的 socket 句柄
  nMsgType　　int，自定义的消息 ID，由插件客户端与服务端协定。
  pMsgData　　BYTE 指针，指向消息数据
  nMsgDataLen　　int，消息数据的字节长度

全局函数

- EIMGetPlgSvrBaseService
  该函数用于获取基础服务接口。

  函数原型：
  __declspec(dllexport)　　IEIMPlgSvrBaseService* EIMGetPlgSvrBaseService();

结构体

- EIMPLGPluginInfo
  该结构体用 PLGPluginInitialize 函数调用时把插件信息传给系统。插件信息包含插件 GUID、插件名称等信息。
  typedef struct tagEIMPLGPluginInfo

```
{
    TCHAR          szPluginGuid[33];
    TCHAR          szPluginName[32];
    TCHAR          szPluginDesc[256];
    TCHAR          szPluginComp[32];
    TCHAR          szPluginUrl[64];
}EIMPLGPluginInfo, *LPEIMPLGPluginInfo;
```

- szPluginGuid　　字符串型，插件 GUID
- szPluginName　　字符串型，插件名称
- szPluginDesc　　字符串型，插件描述
- szPluginComp　　字符串型，插件开发公司名称
- szPluginUrl　　字符串型，插件开发公司网址

第四章 后台管理系统插件

依赖文件

- 头文件：EIMSysMgrPluginServiceInc.h
- 库文件：EIMSysMgrPluginService.lib

插件模块导出函数

一个后台管理系统插件的入口 DLL 必须导出如下三个函数才能被系统加载并被识别为后台管理插件：

- SysMgr_PluginInitialize

插件被初始化时调用的函数，在此函数内插件通过 EIMSysMgrPluginInfo 结构体设置插件 GUID、插件名称、描述等信息，并进行插件内部状态初始化、资源分配等任务。

函数原型：

```
BOOL SysMgr_PluginInitialize(EIMSysMgrPluginInfo& stPluginInfo);
```

- SysMgr_PluginUninitialize

插件被卸载时调用此函数，在此函数内可进行插件自身的资源卸载等任务。

函数原型：

```
BOOL SysMgr_PluginUninitialize();
```

- SysMgr_PluginMsg

后台管理程序通过该函数通知插件有关插件视图显示、隐藏和尺寸变化等事件。插件应当响应这些事件，执行相应的动作。例如，SysMgrHostMsg_ShowView 事件发生时，创建插件的窗体，SysMgrHostMsg_HideView 事件发生时，销毁插件的窗体。

函数原型：

```
BOOL SysMgr_PluginMsg(UINT message, WPARAM wParam, LPARAM lParam);
```

函数参数：

UINT message—指示事件 ID，其值可能是 SysMgrHostMsg_ShowView、SysMgrHostMsg_HideView 或 SysMgrHostMsg_OnSize 等；
WPARAM wParam—消息参数，其意义视消息 ID 而定；
LPARAM lParam—消息参数，其意义视消息 ID 而定。

插件服务接口

- IEIMSysMgrHostService

  该服务接口用于向插件提供一种访问系统数据的能力，目前主要用于从系统中获取当前公司名称和员工简要信息。

  该服务接口通过全局函数 EIMGetSysMgrHostService 获取

  ```
  __declspec(dllexport) IEIMSysMgrHostService*
  EIMGetSysMgrHostService();
  ```

  该服务接口包含如下方法：
  - BOOL GetCompanyName(CString& strCompanyName)
    获取公司名称。公司名称通过strCompanyName返回。

  - BOOL GetStaffInfos(std::vector<LPEIMStaffBriefInfo>& vecStaffInfos)
    获取员工简要信息列表。员工简要信息列表通过vecStaffInfos返回。LPEIMStaffBriefInfo结构体的具体定义见下文结构体部分的描述。

全局函数

- EIMGetSysMgrHostService
  该函数用于获取后台管理系统提供的服务接口

  函数原型：
  ```
  __declspec(dllexport)                    IEIMSysMgrHostService*
  EIMGetSysMgrHostService();
  ```

结构体

- EIMSysMgrPluginInfo
  该结构体用于 SysMgr_PluginInitialize 函数调用时把插件信息传给系统。插件信息包含插件 GUID、插件名称等信息。

```
typedef struct tagEIMSysMgrPluginInfo
{
    TCHAR          szPluginGuid[33];
    TCHAR          szPluginName[32];
    TCHAR          szPluginDesc[256];
    TCHAR          szPluginComp[32];
    TCHAR          szPluginUrl[64];
}EIMSysMgrPluginInfo, *LPEIMSysMgrPluginInfo;
```

- szPluginGuid　　字符串型，插件 GUID
- szPluginName　　字符串型，插件名称
- szPluginDesc　　字符串型，插件描述
- szPluginComp　　字符串型，插件开发公司名称
- szPluginUrl　　字符串型，插件开发公司网址

■ **EIMStaffBriefInfo**

该结构体用于通过 IEIMSysMgrHostService 接口的 GetStaffInfos 方法返回员工的简单描述信息。

```
typedef struct tagEIMStaffBriefInfo
{
    CString        m_strStaffID;
    CString        m_strName;
    CString        m_strSignature;
    IMUserSex      m_emSex;
}EIMStaffBriefInfo, *LPEIMStaffBriefInfo;
```

- m_strStaffID　　字符串型，员工 ID
- m_strName　　字符串型，员工名字
- m_strSignature　　字符串型，员工签名
- m_emSex　　枚举类型，员工性别

消息

SysMgr_PluginMsg 函数的消息及其参数的意见见下表：

消息（message）	WPARAM wParam	LPARAM lParam
● SysMgrHostMsg_ShowView	插件容器窗口句柄	Unused
● SysMgrHostMsg_HideView	插件容器窗口句柄	Unused
● ysMgrHostMsg_OnSize	插件容器窗口句柄	Unused