恒誉短信平台设计说明

目录

[恒誉短信平台设计说明 1](#_Toc357195447)

[一、 引言 2](#_Toc357195448)

[1. 背景 2](#_Toc357195449)

[2. 选择 2](#_Toc357195450)

[二、 概述 3](#_Toc357195451)

[三、 架构设计 4](#_Toc357195452)

[四、 详细设计 5](#_Toc357195453)

[1. 服务端 5](#_Toc357195454)

[a) GPRS Module 5](#_Toc357195455)

[b) Socket Server 7](#_Toc357195456)

[c) SMS Service 9](#_Toc357195457)

[d) SMS Console 10](#_Toc357195458)

[2. 客户端 13](#_Toc357195459)

# 引言

本文档说明了恒誉短信平台的设计理念，软件架构和详细模块。

## 背景

手机短信的行业应用越来越广泛，它具备快捷，经济，针对性强，可移动等优点，很多行业的企业用户都希望能够利用移动（或联通）公司提供的手机短信服务功能，来发布企业信息。如：客户联络、新产品通知、会议通知、紧急通知、催费通知、发货通知、订单查询、节日问候等，从而达到提高办公效率、降低办公成本的目的。

由于短信成为了非常受人欢迎的通信方式，国内各行业信息系统越来越希望能够集成短信收发功能，使整个软件系统功能更丰富、通信成本更低、使用更加便捷，给用户带来更多的利益，从而更加具有竞争力。

## 选择

目前这种有两种方式支持短信群发这个业务，一个就是通过短信猫（工业手机）发送，还有一种就是通过运营商短信网关发送。两种方式各有优缺点：

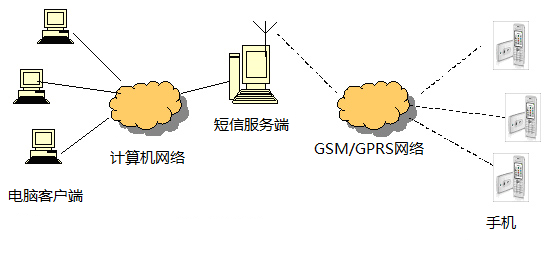
短信猫方案优点是使用简单， 只需要一个短信猫硬件， 不需要联网；缺点是号码是普通手机号码， 性能上也和普通手机差不多， 不适合大规模群发。

短信网关方案优点是高性能高并发， 显示号码可以是运营商的特殊号码； 缺点是需要联网， 需要特殊申请，价格也不便宜。

本短信平台采用短信猫方式。

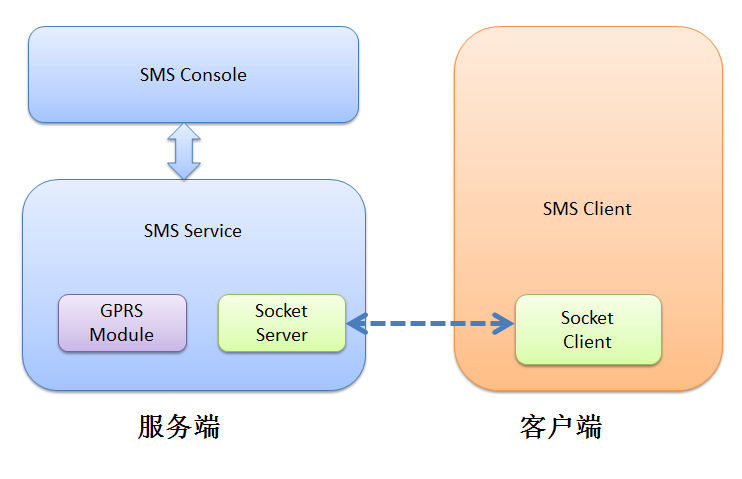


# 概述



本短信平台采用C/S模式， 短信猫硬件安装在服务器端，当短信服务启动之后，可以同时支持多个电脑客户端与之相连， 服务端和客户端通过socket通信。服务端支持短信中心号码设置，端口配置和账号管理等，客户端支持短信收发，通讯簿和短语库等。

# 架构设计



本短信平台采用模块化设计思想，各模块功能独立，可以重复使用。

* **服务端组件**

GPRS Module：实现通过短信猫硬件实现短信收发功能

Socket Server：实现Socket 服务器通讯功能

SMS Service: 将GPRS Module和Socket Server 组成短信后台服务主程序SmsServer.exe

SMS Console: 服务控制台程序SmsConsole.exe, 通过控制SmsServer.exe实现服务的启动，停止，短信服务参数配置和账号管理等。

* **客户端组件**

Socket Client: 实现Socket客户端连接和通讯功能

SMS Client：实现短信收发客户端的主界面SmsExpress.exe, 内部包含通讯簿和短语库，后台以Access为联系人和短信存储数据库

# 详细设计

## 服务端

### GPRS Module

GPRS模块通过串口连接短信猫， 然后通过AT指令操作短信猫硬件，控制短信收发。 下面是基本短信相关的AT指令：

厂家认证 AT+CGMI 获得厂家的标识  
模式认证 AT+CGMM 查询支持频段  
TE设置 AT+CSCS 选择支持网络  
卡的认证 AT+CCID 查询SIM卡的序列号  
功能列表 AT+GCAP 查询可供使用的功能列表  
信号质量 AT+CSQ 查询信号质量  
选择短消息服务 AT+CSMS 选择是否打开短消息服务以及广播服务  
短消息存储 AT+CPMS 选择短消息优先存储区域  
短消息格式 AT+CMGF 选择短消息支持格式（TEXT or PDU）  
保存设置 AT+CSAS 保存+CSCA and +CSMP参数设置  
恢复设置 AT+CRES 恢复+CSCA and +CSMP参数设置  
显示TEXT参数 AT+CSDH 显示当前TEXT模式下结果代码  
新消息提示 AT+CNMI 选择当有新的短消息来时系统提示方式  
读短消息 AT+CMGR 读取短消息  
列短消息 AT+CMGL 将存储的短消息列表  
发送短消息 AT+CMGS 发送短消息  
写短消息 AT+CMGW 写短消息并保存在存储器中  
从内存中发短消息 AT+CMSS 发送在存储器中保存的短消息  
设置TEXT参数 AT+CSMP 设置在TEXT模式下条件参数  
删除短消息 AT+CMGD 删除保存的短消息

下面是部分关键代码：

class CSerialPortEx

{

//function

public:

CSerialPortEx(CSmsComPortAgentWnd\* p, CGPRSManager\* pManager);

~CSerialPortEx(void);

//打开串口

bool initPort(UINT portnr=1,UINT baud=9600,char parity='N',UINT databits=8,UINT stopbits=1);

//关闭串口

void releasePort();

//串口写入数据

void WriteToPort(BYTE\* p,int nLen);

//启动监视线程

bool StartMonitoring();

//停止监视线程

void StopMonitoring();

bool isPortOpen(){return m\_bOpen;}

bool isMonitorRunning(){return m\_bRunning;}

public:

CSmsComPortAgentWnd\* m\_pAgentWnd;

protected:

//串口监视线程

static UINT \_\_stdcall CommThread(LPVOID pParam);

//读取数据

static void ReceiveChar(CSerialPortEx\* port,COMSTAT& comstat);

//写入数据

static void WriteChar(CSerialPortEx\* port);

private:

CSerialPortEx(); //防止缺省构造

unsigned long m\_hThread;

CRITICAL\_SECTION m\_csCommunicationSync;

HANDLE m\_hShutdownEvent;

HANDLE m\_hComm;

HANDLE m\_hWriteEvent;

HANDLE m\_hEventArray[3];

OVERLAPPED m\_ov;

COMMTIMEOUTS m\_CommTimeouts;

BYTE m\_szWriteBuffer[PORT\_WRITE\_BUFFER\_SIZE];//写缓冲

bool m\_bRunning; //线程是否正在运行

bool m\_bOpen; //串口是否已经初始化

DCB m\_dcb;

int m\_nToSend; //需要发送的信息的长度

};

### Socket Server

Socket Server我们采用Select模型， 下面是部分关键代码：

class ISmsCommunicate

{

public:

virtual ~ISmsCommunicate() {}

virtual BOOL InitModule() = 0;

virtual VOID ReleaseModule() = 0;

virtual BOOL SetParameters(LPCTSTR szParams) = 0;

virtual BOOL StartCommunicate() = 0;

virtual BOOL SendMsg(CSMSMessage\* pMsg) = 0;

virtual VOID StopCommunicate() = 0;

virtual INT GetLastErrorCode() = 0;

virtual BOOL IsRunning() = 0;

virtual BOOL CheckIsConnected(GUID& clientId) = 0;

};

class CSmsCommunicateBase: public ISmsCommunicate

{

public:

CSmsCommunicateBase();

virtual ~CSmsCommunicateBase();

virtual BOOL InitModule();

virtual VOID ReleaseModule();

virtual BOOL SetParameters(LPCTSTR szParams) = 0;

virtual BOOL StartCommunicate() = 0;

virtual BOOL SendMsg(CSMSMessage\* pMsg) = 0;

virtual VOID StopCommunicate() = 0;

virtual INT GetLastErrorCode() = 0;

virtual BOOL IsRunning() = 0;

virtual VOID CheckIsAllConnected() = 0;

virtual BOOL CheckIsConnected(GUID& clientId) = 0;

VOID SendMessageToAgent(UINT nMsgType, WPARAM wParam, LPARAM lParam);

protected:

friend class CSmsCommunicateAgentWnd;

VOID OnCommunicateEvent(UINT nMsgType,WPARAM wParam, LPARAM lParam);

protected:

ISMSCommunicateSink\* m\_pCommuSink;

CSmsCommunicateAgentWnd\* m\_pAgentWnd;

};

class CSmsSocketCommunicate:public CSmsCommunicateBase

{

public:

CSmsSocketCommunicate(ISMSCommunicateSink\* pSink);

virtual ~CSmsSocketCommunicate();

virtual BOOL SetParameters(LPCTSTR szParams);

virtual BOOL StartCommunicate();

virtual BOOL SendMsg(CSMSMessage\* pMsg);

virtual VOID StopCommunicate();

virtual INT GetLastErrorCode();

virtual BOOL IsRunning();

virtual BOOL CheckIsConnected(GUID& clientId);

VOID CheckIsAllConnected();

protected:

BOOL AddConnection(CSmsSocket\*);

BOOL RemoveConnection(SOCKET socket);

VOID OnReceive(SOCKET socket,SMS\_SOCKET\_HEADER\* header);

static UINT monitor(LPVOID);

UINT myWrapMonitor();

VOID RemoveAll();

private:

CSimpleArray<CSmsSocket\*> m\_arrSocket;

fd\_set m\_fd;

HANDLE m\_hThread;

INT m\_nErrorCode;

unsigned short m\_nPort;

BOOL m\_bRunning;

HANDLE m\_hStartEvent;

CRITICAL\_SECTION m\_cs;

};

### SMS Service

SMS Service是短信收发和Socket通讯的后台服务程序， 下面是它部分关键代码：

class CSmsServiceMgr:public ISMSCommunicateSink,public ISMSDeviceSink

{

public :

CSmsServiceMgr(HWND hWnd);

virtual ~CSmsServiceMgr();

BOOL Start();

VOID Stop();

VOID FeedbackMessageToController(UINT nMsgType,WPARAM wParam, LPARAM lParam, BOOL bPost = TRUE);

VOID ServiceStatusNotify();

VOID ProcessControllerRequest(UINT, WPARAM, LPARAM);

VOID CheckIsAccountUsing();

//ISMSCommunicateSink

virtual VOID OnCommunicateLogon(CSMSMessage\* pMsg);

virtual VOID OnCommunicateSend(CSMSMessage\* pMsg);

virtual VOID OnCommunicateDisconnect(GUID& clientId);

virtual VOID OnCommunicateStarted();

virtual VOID OnCommunicateStopped();

virtual VOID OnCommunicateError(INT\* pReason);

//ISMSDeviceSink

virtual VOID OnDeviceReceiveMsg(CSMSMessage\* msg);

virtual VOID OnDeviceSendedMsg(CSMSMessage\* msg);

virtual VOID OnDeviceStarted();

virtual VOID OnDeviceStopped();

virtual VOID OnDeviceError(INT\* pReason);

virtual VOID OnDeviceSignalCount(INT\* pCount);

BOOL IsRunning();

private:

CSmsFactory& m\_SmsFactory;

ISmsDevice\* m\_pSmsDevice;

ISmsCommunicate\* m\_pSmsCommunicate;

ISmsConfig\* m\_pSmsConfig;

CSmsAccountManager\* m\_pSmsAccountManager;

HWND m\_hController;

INT m\_nErrorCode;

// BOOL m\_bStarting;

};

### SMS Console

SMS Console是短信图形化控制台程序，它可以启动，停止和配置短信服务参数。下面是部分关键代码：

class CMainDlg : public CDialogImpl<CMainDlg>, public CUpdateUI<CMainDlg>,

public CMessageFilter, public CIdleHandler,public CDialogSkin

{

public:

enum { IDD = IDD\_MAINDLG };

enum {ID\_PANE\_STATUS = 1, ID\_PANE\_COUNT = 2};

enum {PANE\_WIDTH = 310, COUNT\_WIDTH = 50, STATUS\_WIDTH = 250};

enum ENUM\_UI\_STATUS {UI\_STATUS\_STOPPED, UI\_STATUS\_RUNNING,UI\_STATUS\_STARTING};

CMainDlg();

~CMainDlg();

virtual BOOL PreTranslateMessage(MSG\* pMsg)

{

if(m\_pSettingDlg->IsWindow())

{

return m\_pSettingDlg->IsDialogMessage(pMsg);

}

return IsDialogMessage(pMsg);

}

virtual BOOL OnIdle()

{

return FALSE;

}

BEGIN\_UPDATE\_UI\_MAP(CMainDlg)

END\_UPDATE\_UI\_MAP()

BEGIN\_MSG\_MAP(CMainDlg)

MESSAGE\_HANDLER(WM\_MOUSEMOVE, OnMouseMove)

MESSAGE\_HANDLER(WM\_INITDIALOG, OnInitDialog)

MESSAGE\_HANDLER(WM\_CLOSE, OnClose)

MESSAGE\_HANDLER(WM\_DESTROY, OnDestroy)

// MESSAGE\_HANDLER(WM\_COPYDATA,OnCopyData)

MESSAGE\_HANDLER(WM\_PAINT, OnPaint)

MESSAGE\_HANDLER(WM\_SHELL\_MSG, OnTrayMsg)

MESSAGE\_HANDLER(WM\_SMS\_ON\_STOP,OnStopMessage)

MESSAGE\_HANDLER(WM\_SMS\_ON\_START,OnStartMessage)

MESSAGE\_HANDLER(WM\_SMS\_ON\_ERROR,OnErrorMessage)

MESSAGE\_HANDLER(WM\_SMS\_ON\_DEVICE\_SIGNAL, OnGetSignal)

MESSAGE\_HANDLER(WM\_SMS\_ON\_CONNECT\_COUNT, OnGetConCount)

MESSAGE\_HANDLER(WM\_TIMER, OnTimer)

// MESSAGE\_HANDLER(WM\_SMS\_ON\_STATUS\_INDICATION,OnStatusIndication)

MESSAGE\_HANDLER(WM\_SMS\_ON\_RECEIVE\_MESSAGE, OnReceiveMessage)

MESSAGE\_HANDLER(WM\_SMS\_ON\_SEND\_MESSAGE, OnSendedMessage)

MESSAGE\_HANDLER(WM\_SMS\_ON\_CONNECT, OnSmsConnect)

MESSAGE\_HANDLER(WM\_SMS\_ON\_DISCONNECT, OnSmsDisconnect)

COMMAND\_ID\_HANDLER(IDC\_START, OnStart)

COMMAND\_ID\_HANDLER(IDC\_STOP, OnStop)

COMMAND\_ID\_HANDLER(IDC\_SETTING, OnSetting)

COMMAND\_ID\_HANDLER(ID\_SHOW\_MAIN, OnShowMainDlg)

COMMAND\_ID\_HANDLER(ID\_ABOUT, OnAbout)

COMMAND\_ID\_HANDLER(ID\_EXIT, OnExit)

CHAIN\_MSG\_MAP(CDialogSkin)

REFLECT\_NOTIFICATIONS()

END\_MSG\_MAP()

// Handler prototypes (uncomment arguments if needed):

// LRESULT MessageHandler(UINT /\*uMsg\*/, WPARAM /\*wParam\*/, LPARAM /\*lParam\*/, BOOL& /\*bHandled\*/)

// LRESULT CommandHandler(WORD /\*wNotifyCode\*/, WORD /\*wID\*/, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/)

// LRESULT NotifyHandler(int /\*idCtrl\*/, LPNMHDR /\*pnmh\*/, BOOL& /\*bHandled\*/)

LRESULT OnTimer(UINT nMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnInitDialog(UINT /\*uMsg\*/, WPARAM /\*wParam\*/, LPARAM /\*lParam\*/, BOOL& /\*bHandled\*/);

LRESULT OnPaint(UINT nMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnGetSignal(UINT uMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnGetConCount(UINT, WPARAM wParam, LPARAM, BOOL&);

// LRESULT OnCopyData(UINT uMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

// LRESULT OnStatusIndication(UINT nMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnStartMessage(UINT nMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnStopMessage(UINT nMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnErrorMessage(UINT nMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnStart(WORD /\*wNotifyCode\*/, WORD wID, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnStop(WORD /\*wNotifyCode\*/, WORD wID, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnMouseMove(UINT nMsgType,WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnDestroy(UINT nMsgType , WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnClose(UINT nMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnReceiveMessage(UINT nMsgType, WPARAM wParam, LPARAM lPram, BOOL& bHandled);

LRESULT OnSendedMessage(UINT nMsgType, WPARAM wParam, LPARAM lPram, BOOL& bHandled);

LRESULT OnSetting(WORD /\*wNotifyCode\*/, WORD wID, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnSmsConnect(UINT, WPARAM, LPARAM, BOOL&);

LRESULT OnSmsDisconnect(UINT, WPARAM, LPARAM, BOOL&);

LRESULT OnTrayMsg(UINT, WPARAM, LPARAM, BOOL&);

LRESULT OnShowMainDlg(WORD, WORD, HWND, BOOL&);

LRESULT OnAbout(WORD, WORD ,HWND ,BOOL&);

LRESULT OnExit(WORD, WORD, HWND, BOOL&);

HINSTANCE GetResourceInstance() {return m\_hResModule;}

// CSmsAccountManager\* GetAccountManager() {return m\_pAccountManager;}

BOOL IsServiceRunning() {return (FindAgentWnd() != NULL);}

VOID ResetCaption();

private:

VOID PaintCurrentInformation(CPaintDC& dc);

HWND FindAgentWnd();

BOOL KillAgentExist();

BOOL Stop();

BOOL Start();

VOID SetUIStatus(ENUM\_UI\_STATUS s);

VOID ResetValues();

private:

CIEButton m\_btnStart;

CIEButton m\_btnStop;

CIEButton m\_btnSetting;

// HICON m\_hStartingIcon;

// HICON m\_hStoppedIcon;

// HICON m\_hRunningIcon;

HINSTANCE m\_hResModule;

CMultiPaneStatusBarCtrl m\_statusbar;

//CSmsAccountManager\* m\_pAccountManager;

ENUM\_UI\_STATUS m\_eCurrentStatus;

HWND m\_hServiceAgent;

CSetupDlg\* m\_pSettingDlg;

TCHAR m\_szErrorMsg[MAX\_PATH];

CConsoleTray\* m\_pConsoleTray;

INT m\_nConCount;

INT m\_nSignalCount;

INT m\_nSendedCount;

INT m\_nDispatchedCount;

BOOL m\_bExit;

};

## 客户端

短信客户端实现短信收发客户端的主界面SmsExpress.exe, 内部包含通讯簿和短语库，后台以Access为联系人和短信存储数据库。

下面是部分关键代码：

class CMainFrame : public CFrameWindowImpl<CMainFrame>, public CUpdateUI<CMainFrame>,

public CMessageFilter, public CIdleHandler,

private CToolBarHelper<CMainFrame>

{

public:

enum ENUM\_SMS\_CLIENT\_STATUS

{

STOPPED = 0,

BUSY,

IDLE,

STARTING

};

enum

{

PANE\_INFO = 1,

PANE\_COUNT,

PANE\_COMPANY,

PANE\_STATUS,

PANE\_INFO\_WIDTH = 300,

PANE\_COUNT\_WIDTH = 200,

PANE\_STATUS\_WIDTH = 80,

PANE\_ALL\_MIN\_WIDTH = PANE\_INFO\_WIDTH + PANE\_COUNT\_WIDTH + PANE\_STATUS\_WIDTH

};

DECLARE\_FRAME\_WND\_CLASS(\_T("SmsClientMainFrame\_Richard\_612"), IDR\_MAINFRAME)

CMainFrame(CServiceManager\*);

~CMainFrame();

virtual BOOL PreTranslateMessage(MSG\* pMsg);

virtual BOOL OnIdle();

void OnToolBarCombo(HWND hWndCombo, UINT nID, int nSel, LPCTSTR lpszText, DWORD dwItemData);

virtual void PrepareToolBarMenu(UINT nMenuID, HMENU hMenu);

BEGIN\_UPDATE\_UI\_MAP(CMainFrame)

UPDATE\_ELEMENT(ID\_VIEW\_TOOLBAR, UPDUI\_MENUPOPUP)

UPDATE\_ELEMENT(ID\_VIEW\_STATUS\_BAR, UPDUI\_MENUPOPUP)

UPDATE\_ELEMENT(ID\_SYSTEM\_CONNECT, UPDUI\_MENUPOPUP|UPDUI\_TOOLBAR)

UPDATE\_ELEMENT(ID\_SYSTEM\_DISCONNECT, UPDUI\_MENUPOPUP|UPDUI\_TOOLBAR)

END\_UPDATE\_UI\_MAP()

BEGIN\_MSG\_MAP(CMainFrame)

MESSAGE\_HANDLER(WM\_CREATE, OnCreate)

MESSAGE\_HANDLER(WM\_CLOSE, OnClose)

MESSAGE\_HANDLER(WM\_TIMER, OnTimer)

MESSAGE\_HANDLER(WM\_SIZE, OnSize)

//MESSAGE\_HANDLER(WM\_QUERYENDSESSION, OnQueryEndSession)

MESSAGE\_HANDLER(WM\_MENUSELECT, OnMenuSelect)

MESSAGE\_HANDLER(WM\_VIEW\_CHANGED, OnViewChanged)

MESSAGE\_HANDLER(WM\_SHELL\_MSG, OnTrayMsg)

COMMAND\_ID\_HANDLER(ID\_VIEW\_TOOLBAR, OnViewToolBar)

COMMAND\_ID\_HANDLER(ID\_VIEW\_STATUS\_BAR, OnViewStatusBar)

MESSAGE\_HANDLER(WM\_SMS\_CLIENT, OnSocketClientEvent)

COMMAND\_ID\_HANDLER(ID\_SYSTEM\_CONNECT, OnConnect)

COMMAND\_ID\_HANDLER(ID\_SYSTEM\_DISCONNECT, OnDisconnect)

COMMAND\_ID\_HANDLER(ID\_SYSTEM\_SETTING, OnSetting)

COMMAND\_ID\_HANDLER(ID\_SYSTEM\_EXIT, OnExit)

COMMAND\_ID\_HANDLER(ID\_TOOL\_CONTACTLIST, OnToolContact)

COMMAND\_ID\_HANDLER(ID\_TOOL\_SMSLIB, OnToolLib)

COMMAND\_ID\_HANDLER(ID\_HELP\_ABOUT, OnAbout)

COMMAND\_ID\_HANDLER(ID\_HELP\_QUESTION, OnQuestion)

COMMAND\_ID\_HANDLER(ID\_SYSTEM\_SHOW, OnShowWindow)

CHAIN\_MSG\_MAP(CUpdateUI<CMainFrame>)

CHAIN\_MSG\_MAP(CFrameWindowImpl<CMainFrame>)

CHAIN\_MSG\_MAP(CToolBarHelper<CMainFrame>)

CHAIN\_MSG\_MAP\_ALT\_MEMBER(m\_view, 1)

END\_MSG\_MAP()

// Handler prototypes (uncomment arguments if needed):

// LRESULT MessageHandler(UINT /\*uMsg\*/, WPARAM /\*wParam\*/, LPARAM /\*lParam\*/, BOOL& /\*bHandled\*/)

// LRESULT CommandHandler(WORD /\*wNotifyCode\*/, WORD /\*wID\*/, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/)

// LRESULT NotifyHandler(int /\*idCtrl\*/, LPNMHDR /\*pnmh\*/, BOOL& /\*bHandled\*/)

VOID LoadSms();

LRESULT OnTrayMsg(UINT, WPARAM, LPARAM, BOOL&);

LRESULT OnMenuSelect(UINT, WPARAM, LPARAM, BOOL&);

LRESULT OnViewChanged(UINT, WPARAM, LPARAM, BOOL&);

LRESULT OnSize(UINT, WPARAM, LPARAM, BOOL& bHandled);

LRESULT OnCreate(UINT /\*uMsg\*/, WPARAM /\*wParam\*/, LPARAM /\*lParam\*/, BOOL& /\*bHandled\*/);

LRESULT OnViewToolBar(WORD /\*wNotifyCode\*/, WORD /\*wID\*/, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnViewStatusBar(WORD /\*wNotifyCode\*/, WORD /\*wID\*/, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnConnect(WORD /\*wNotifyCode\*/, WORD /\*wID\*/, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnDisconnect(WORD /\*wNotifyCode\*/, WORD /\*wID\*/, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnSetting(WORD /\*wNotifyCode\*/, WORD /\*wID\*/, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnExit(WORD /\*wNotifyCode\*/, WORD /\*wID\*/, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnToolContact(WORD /\*wNotifyCode\*/, WORD /\*wID\*/, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnToolLib(WORD /\*wNotifyCode\*/, WORD /\*wID\*/, HWND /\*hWndCtl\*/, BOOL& /\*bHandled\*/);

LRESULT OnClose(UINT nMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnTimer(UINT nMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

LRESULT OnAbout(WORD, WORD, HWND, BOOL&);

LRESULT OnShowWindow(WORD, WORD, HWND, BOOL&);

LRESULT OnQuestion(WORD, WORD ,HWND, BOOL&);

//LRESULT OnQueryEndSession(UINT, WPARAM, LPARAM, BOOL&);

LRESULT OnSocketClientEvent(UINT nMsgType, WPARAM wParam, LPARAM lParam, BOOL& bHandled);

static HINSTANCE GetResourceModule() { return m\_hResModule;}

static CMainFrame\* GetMainFrame() {return m\_pMainFrame;}

static CSmsClientView\* GetClientView() {return m\_pClientView;}

CSmsClientConfig\* GetSmsClientConfig() {return m\_pSmsConfig;}

CServiceManager\* GetServiceMgr() {return m\_pServiceManager;}

VOID StartConnect();

VOID Disconnect();

VOID ResetStatusBar();

BOOL SetExitFlag(BOOL b) {m\_bExit = b;}

public:

CCommandBarCtrl m\_CmdBar;

private:

VOID GetErrorMsg(int nError, CString& strMsg);

private:

CSmsClientView m\_view;

CMultiPaneStatusBarCtrl m\_statusBar;

static HINSTANCE m\_hResModule;

static CMainFrame\* m\_pMainFrame;

static CSmsClientView\* m\_pClientView;

CServiceManager\* m\_pServiceManager;

CSmsClientConfig\* m\_pSmsConfig;

INT m\_nTimer;

ENUM\_SMS\_CLIENT\_STATUS m\_eCurrentStatus;

INT m\_nSendedCount;

INT m\_nReceivedCount;

CString m\_strInforStatus;

CSmsTray\* m\_pTray;

BOOL m\_bSaveWinPos;

BOOL m\_bExit;

};