```
#if !defined(AFX_MAINDLG_H__96E11F88_523A_4147_9E48_88B541251266__INCLUDED_)
#define AFX_MAINDLG_H__96E11F88_523A_4147_9E48_88B541251266__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "MyServerPort.h"
#include "MyCounter.h"

// CMainDlg dialog
class CMainDlg : public CDialog
{
// Construction
public:
	CMainDlg(CWnd* pParent = NULL);   // standard constructor
// Dialog Data
	//{{AFX_DATA(CMainDlg)
	enum { IDD = IDD_USAGEMETER_DIALOG };
	CStatic			m_status_lbl;
	CTabCtrl	m_Tab_Ctrl;
	//}}AFX_DATA
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CMainDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	virtual LRESULT WindowProc(UINT message, WPARAM wParam, LPARAM lParam);
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;
	// Generated message map functions
	//{{AFX_MSG(CMainDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnSelchangingTab(NMHDR* pNMHDR, LRESULT* pResult);
	virtual void OnOK();
	afx_msg void OnTimer(UINT nIDEvent);
	afx_msg void OnClose();
	afx_msg void OnFileViewer();
	afx_msg void OnBeepSet();
	#ifdef _DEBUG
	afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
	#endif
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
//********************** CUSTOM *****************************
private:
	//Variables
	char   m_temp_duration[9];
	char   m_win_str_time[9];
	char   m_int_str_time[9];
	char   m_res_zerotime[9];
	char   m_temp_rugh_dura[9];
	char   m_now[9];
	char   m_res_tip_text[106];
	char   m_res_label_text[135];
	char   *m_win_duration,*m_int_duration;
	char   *m_win_total,*m_int_total;
	int           m_select_tab;
	int           m_win_str_sec, m_int_str_sec;
	HWND   m_hint_list, m_hwin_list;
	CWnd*  m_about_dlg;
	DWORD  m_privilage;
	CMenu* m_pop_menu;
	HFONT  m_font_win, m_font_int;
	CMyCounter          m_win_counter,m_int_counter;
	CMyServerPort m_my_api;
	//Functions
```

```
	#ifdef _DEBUG
		bool	Fake_Connection();
	#endif
		void	KillMeter();
		DWORD	FindLatsSlash(char *path,DWORD len);
inline void	SetTip(int mode,int prayority);
inline void	SetLable(int mode);
		void	SetUpTray(DWORD Task,char* tip,int size);
		void	SetBeep(bool reset_int_count, bool reset_win_count);
		void	StartUp();
		void	TimerMain();
		void	ShutDown();
		void	SetUpDialog();
		void	SaveWindowPos();
		void	ShowAboutBox();
		void	GetAppPath(char* path,DWORD* size, bool writerunkey);
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.

#endif // !defined(AFX_MAINDLG_H__96E11F88_523A_4147_9E48_88B541251266__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

//************ 10-07-2004 18:49 ****************
#include "stdafx.h"
#include "Usage Meter.h"
#include "MainDlg.h"
#include "DlgBeepSetting.h"
#include "MyListView.h"
#include <Wininet.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// CAboutDlg dialog used for App
About*****************************************************
class CAboutDlg : public CDialog
{
public:
	CAboutDlg();
	CWnd** m_my_handle;//pointer to CMainDlg CWnd* m_about_dlg
// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL
// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnClose();
	virtual void OnOK();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

// CMainDlg
dialog*********************************************************************************
CMainDlg::CMainDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMainDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMainDlg)
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
```

```
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
void CMainDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CMainDlg)
	DDX_Control(pDX, IDC_STATUS, m_status_lbl);
	DDX_Control(pDX, IDC_TAB, m_Tab_Ctrl);
	//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CMainDlg, CDialog)
	//{{AFX_MSG_MAP(CMainDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_NOTIFY(TCN_SELCHANGING, IDC_TAB, OnSelchangingTab)
	ON_WM_TIMER()
	ON_WM_CLOSE()
	ON_BN_CLICKED(ID_FILE_VIEWER, OnFileViewer)
	ON_BN_CLICKED(ID_BEEP_SET, OnBeepSet)
	ON_WM_LBUTTONUP()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CMainDlg message handlers
BOOL CMainDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);
	/*CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{	CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}*/
	SetIcon(m_hIcon, TRUE);
	SetIcon(m_hIcon, FALSE);

	StartUp();

	return TRUE;
}
//-----------------------------------------------------------------------------------------
void CMainDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		ShowAboutBox();
	}else{
		CDialog::OnSysCommand(nID, lParam);
	}
}
void CMainDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting
		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;
		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}else{
```

```
		CDialog::OnPaint();
	}
}
HCURSOR CMainDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}
//----------------------------------------------------------------------------------
LRESULT CMainDlg::WindowProc(UINT message, WPARAM wParam, LPARAM lParam)
{
switch(message)
{
case ID_TRAY_ICON:
//Tray icon Msg
	if(lParam==WM_LBUTTONDOWN)
	{//LBUTTON DOWN
		ShowWindow(1);
		SetForegroundWindow();
		return 0;
	}
	if(lParam==WM_RBUTTONDOWN)
	{//LBUTTON DOWN
		delete m_pop_menu;
		m_pop_menu = new CMenu();
		m_pop_menu->CreatePopupMenu();
		if(IsWindowVisible())
		{
		m_pop_menu->AppendMenu( MF_STRING, ID_TRAY_SHOEHIDE,"Hide" );
		}else{
		m_pop_menu->AppendMenu( MF_STRING, ID_TRAY_SHOEHIDE,"Show" );
		}
		m_pop_menu->SetDefaultItem(0,TRUE);
		m_pop_menu->AppendMenu( MF_STRING, ID_FILE_VIEWER,"File Viewer" );
		m_pop_menu->AppendMenu( MF_SEPARATOR, NULL,"");
		m_pop_menu->AppendMenu( MF_STRING, ID_TRAY_ABOUT,"About..." );
		CPoint cur_point;
		GetCursorPos(&cur_point);
		SetForegroundWindow();
		m_pop_menu->TrackPopupMenu( TPM_LEFTALIGN | TPM_RIGHTBUTTON,cur_point.x,
cur_point.y,this);
		return 0;
	}
break;

case WM_COMMAND:
	switch(wParam)
	{//Tray menu commands
	case ID_TRAY_ABOUT:
		ShowAboutBox();
	break;
	case ID_TRAY_SHOEHIDE:
		if(IsWindowVisible()){
		ShowWindow(0);
		}else{
		ShowWindow(1);}
	break;
	}
break;

case WM_ENDSESSION:
//ShutDown,LogOff
	ShutDown();
}
	return CDialog::WindowProc(message, wParam, lParam);
}
//----------------------------------------------------------------------------------
void CMainDlg::OnClose()
{
	ShowWindow(0);
}
//----------------------------------------------------------------------------------
void CMainDlg::OnOK()
{
```

```
	ShowWindow(0);
}
//--------------------------------------------------------------------------------------
void CMainDlg::OnSelchangingTab(NMHDR* pNMHDR, LRESULT* pResult)
{//Tab changed
	if(m_Tab_Ctrl.GetItemState(1,1) == 1)
	{//Internet
	::ShowWindow(m_hwin_list,0);
	::ShowWindow(m_hint_list,1);
	m_select_tab = 0;
	}
	else
	{//Windows
	::ShowWindow(m_hwin_list,1);
	::ShowWindow(m_hint_list,0);
	m_select_tab = 1;
	};

	//TimerMain();
	*pResult = 0;
}
//--------------------------------------------------------------------------------------
void CMainDlg::OnTimer(UINT nIDEvent)
{//Timer Msges
int		x,y;
static BYTE  i;
	switch (nIDEvent)
	{
	case TIMER_MAIN:
		TimerMain();
		break;
	case TIMER_HIDE:
		if(i==2)//Hide timer ticks 3 times to ensure the mighty Usage Meter is hidden(ha...
ha...)
			KillTimer(TIMER_HIDE);
		i++;
		ShowWindow(0);
		//Posision Main Dlg
		x= m_my_api.RegGet(HKEY_CURRENT_USER,"x");
		y= m_my_api.RegGet(HKEY_CURRENT_USER,"y");
		if (x==0||y==0){ x=100 ;y=100;}
		SetWindowPos(&CWnd::wndTop,x,y,0,0,SWP_NOSIZE);
		break;
	case TIMER_SERVER_LINK:
		//chek wether a server given privilage
		m_my_api.GrantPrivilages(this->m_hWnd);
		break;
	}
CDialog::OnTimer(nIDEvent);
}
//--------------------------------------------------------------------------------------
void CMainDlg::SetUpDialog()
{
CRect dlg_rect;
CWnd* static_ctrl;

	//Add about item to system menu
	HMENU hsys_menu = ::GetSystemMenu(this->m_hWnd,FALSE);
	::AppendMenu(hsys_menu,MF_SEPARATOR,NULL,NULL);
	::AppendMenu(hsys_menu,MF_STRING, IDM_ABOUTBOX,"About...");

	GetClientRect(&dlg_rect);

	//Tab Cntl
	m_Tab_Ctrl.InsertItem(0,"Internet");
	m_Tab_Ctrl.InsertItem(1,"Windows");
	m_Tab_Ctrl.MoveWindow(5,6,dlg_rect.Width()-6,dlg_rect.Height()-40);
	//Static
	static_ctrl=GetDlgItem(IDC_STATIC);
	static_ctrl->SetWindowPos(&CWnd::wndTop ,10,32,dlg_rect.Width()-18,dlg_rect.Height()-
73,SWP_SHOWWINDOW);
	//Create ListView Fonts
LOGFONT lf;
```

```
	memset(&lf, 0, sizeof(LOGFONT));
	lf.lfHeight = 13;
	strcpy(lf.lfFaceName, "Verdana");
	m_font_int = ::CreateFontIndirect(&lf);
	m_font_win = ::CreateFontIndirect(&lf);
	//SetUp listView
CMyListView list;
	list.SetupListview(m_hwin_list,m_hint_list,m_font_win,m_font_int);

	//Hide win list
	::ShowWindow(m_hwin_list,0);
}
//--------------------------------------------------------------------------------------------
void CMainDlg::SetUpTray(DWORD Task,char* tip,int size)
{
NOTIFYICONDATA nd;
	nd.cbSize=sizeof(nd);
	nd.uCallbackMessage=ID_TRAY_ICON;
	nd.hIcon=CDialog::GetIcon(false);
	nd.hWnd=this->m_hWnd;
	nd.uID=1;
	nd.uFlags=NIF_ICON | NIF_MESSAGE | NIF_TIP ;
	strncpy(nd.szTip, tip ,size);
	Shell_NotifyIcon(Task, &nd);
}
//--------------------------------------------------------------------------------------------
void CMainDlg::SaveWindowPos()
{
CRect dlg_rect;
	//Save pos
	GetWindowRect(&dlg_rect);
	m_my_api.RegSet(HKEY_CURRENT_USER,"x",dlg_rect.left);
	m_my_api.RegSet(HKEY_CURRENT_USER,"y",dlg_rect.top);
}
//--------------------------------------------------------------------------------------------
void CMainDlg::ShowAboutBox()
{
	if(m_about_dlg==NULL)
	{
		CAboutDlg dlgAbout;
		dlgAbout.m_my_handle = &m_about_dlg;
		dlgAbout.DoModal();
	}
	else
	{
		m_about_dlg->SetForegroundWindow();
	}
}
//--------------------------------------------------------------------------------------------
void CMainDlg::ShutDown()
{
	delete m_pop_menu;
	KillTimer(TIMER_SERVER_LINK);
	KillTimer(TIMER_MAIN);
	SetUpTray(NIM_DELETE,NULL,0);
	SaveWindowPos();

	if(m_privilage==SERVER)
	{//Give privilage to running client to be a server
		if(m_my_api.GivePrivilage()==false)
		{//no clients to get privilage so I should save log
			//Remove connection status file
			m_my_api.SetStatusFile(CLEAR_STATUS);
			m_my_api.GetTime(m_now);
			DAY_STRUCT today;
			m_my_api.GetDay(&today);
			//save windows usage
			m_my_api.TimeElapse(m_win_str_sec,m_now,m_temp_duration);
			m_win_counter.GetRoughDuration(m_temp_rugh_dura);

	m_my_api.Save_Win_Usage(&today,m_win_str_time,m_now,m_temp_duration,m_temp_rugh_dura);
```

```
                        //(save internet usage) This is only done if currently connected this
connection will
                        //terminate shortly after UsageMeter diminish this situation ocour when user
log off
                        //without disconnecting the internet
                DWORD contype;
                        if(InternetGetConnectedState(&contype, NULL))
                        {
                                if((contype&INTERNET_CONNECTION_MODEM)==INTERNET_CONNECTION_MODEM)
                                {
                                m_my_api.TimeElapse(m_int_str_sec,m_now,m_temp_duration);
                                m_int_counter.GetRoughDuration(m_temp_rugh_dura);

        m_my_api.Save_Int_Usage(&today,m_int_str_time,m_now,m_temp_duration,m_temp_rugh_dura);
                                }
                        }
                }
        }

        ::DeleteObject(m_font_win);
        ::DeleteObject(m_font_int);
        m_my_api.RemoveUser();
}
//--------------------------------------------------------------------------------------
void CMainDlg::StartUp()
{
//common routine for any mode
        //Get ListView Handles
        m_hwin_list = ::GetDlgItem(this->m_hWnd,IDC_WIN_LIST);
        m_hint_list = ::GetDlgItem(this->m_hWnd,IDC_INT_LIST);
        //Set counters reference
        m_win_counter.m_pstart_sec = &m_win_str_sec;
        m_int_counter.m_pstart_sec = &m_int_str_sec;
        //Check beep settings and set beeps
        SetBeep(true,true);
        //set popup menu pointer to NULL (it's safer)
        m_about_dlg = NULL;
        m_pop_menu = NULL;
        m_select_tab = 0;
        //Load resources
        memcpy(m_res_zerotime,"00:00:00\0",9);
        memcpy(m_temp_duration,m_res_zerotime,9);
        memcpy(m_temp_rugh_dura,m_res_zerotime,9);
        memcpy(m_win_str_time,m_res_zerotime,9);
        memcpy(m_int_str_time,m_res_zerotime,9);
        //Fill m_res_tip_text buffer (Bit tricky)**********
        memcpy(m_res_tip_text,"Internet Usage : 000:00:00\n\rWindows running : 00:00:00\0",55);
        m_win_duration = m_res_tip_text + 46;//Directly send a part of buffer to counters
(Efficiancy man Efficiancy...)
        m_int_total    = m_res_tip_text + 17;
        memcpy(m_res_tip_text+56,"Connected : 00:00:00\n\rWindows running : 00:00:00\0",49);
        m_int_duration = m_res_tip_text+68;
        //Fill m_res_label_text buffer
        memcpy(m_res_label_text,"Offline     |     Total - 000:00:00\0",36);
        memcpy(m_res_label_text+37,"Connected - 00:00:00     |     Total - 000:00:00\0",49);
        memcpy(m_res_label_text+86,"Running - 00:00:00     |     Total - 000:00:00\0",47);
        m_win_total = m_res_label_text + 123;
        //initialize ServerPortObj
        m_my_api.IntServerPortObj(&m_privilage);
        //Write Run key
char    temp_path[_MAX_PATH];
DWORD   temp_len = _MAX_PATH;
        GetAppPath(temp_path,&temp_len,true);
        //Set up Dialog box
        SetUpDialog();
        //Set Tray Icon
        SetUpTray(NIM_ADD," Usage Meter \0",14);
        //Posision window away from screen
        RECT   r;
        ::GetWindowRect(::GetDesktopWindow(),&r);
        ::SetWindowPos(this->m_hWnd, HWND_BOTTOM,r.right+10,0,0,0,SWP_NOSIZE);
        //check server running
DWORD   temp_winroughdu;
```

```
	m_my_api.IsServerRunning(m_win_str_time,&temp_winroughdu);
	//initialize LogFileObj
	m_my_api.IntLogFileObj(m_hint_list,m_hwin_list,m_int_total,m_win_total,this,&m_privilage);
	//Load log
	if(m_my_api.LoadLog()==false)
	{//Cannot load the log file
		KillMeter();
		return;
	}

	if (m_privilage==SERVER)
	{//fresh start
	//Remove connection status file "Connected"
	m_my_api.SetStatusFile(DISCONNECTED);
	m_my_api.InitializeUserList();
	//Save windows start time
	m_my_api.GetTime(m_win_str_time);
	m_win_counter.Reset();
	}

	//At this point m_win_str_time can conver to seconds
	m_win_str_sec = m_my_api.ToSeconds(m_win_str_time);

	if (m_privilage==CLIENT)
	{
	//add current user to the registry
	m_my_api.AddUser();
	//Calculat win duration
	m_my_api.GetTime(m_now);
	m_my_api.TimeElapse(m_win_str_sec,m_now,m_temp_duration);
	//Set starting poit for win counter
	m_win_counter.SetAt(m_temp_duration,temp_winroughdu,true);
	//I am a client client timer tick faster for comunication efficiency
	SetTimer(TIMER_SERVER_LINK,500,0);
	}

	//Start main timer
	SetTimer(TIMER_MAIN,1000,0);
	SetTip(ID_MODE_DESCONECT , ID_TIP_PRAYORITY_HIGH);
	//start hide timer
	SetTimer(TIMER_HIDE,200,0);
}
//-----------------------------------------------------------------------------
void CMainDlg::SetLable(int mode)
{
	if(m_select_tab == 1)
	{//Windows tab selected
		memcpy(m_res_label_text+96,m_win_duration,8);
		m_status_lbl.SetWindowText(m_res_label_text+86);
	}else{
	//Internet tab selected
		if(mode==ID_MODE_CONECT)
		{
			memcpy(m_res_label_text+49,m_int_duration,8);
			memcpy(m_res_label_text+76,m_int_total,9);
			m_status_lbl.SetWindowText(m_res_label_text+37);
		}else{
			memcpy(m_res_label_text+26,m_int_total,9);
			m_status_lbl.SetWindowText(m_res_label_text);
		}
	}
}
//-----------------------------------------------------------------------------
void CMainDlg::SetTip(int mode,int prayority)
{
static int count = 0;

	if(prayority!=ID_TIP_PRAYORITY_HIGH)
	{	if(count<5)
		{count++;
		return;}
	}
	count=0;
```

```
	if(mode==ID_MODE_CONECT)
	{
		memcpy(m_res_tip_text+96,m_win_duration,8);
		SetUpTray(NIM_MODIFY,m_res_tip_text+56,50);
	}else{
		SetUpTray(NIM_MODIFY,m_res_tip_text,55);
	}
}
//--------------------------------------------------------------------------------
void CMainDlg::TimerMain()
{
#ifdef _DEBUG
bool		connected;
#else
BOOL		connected;
#endif

DAY_STRUCT today;
char	czprevi_day[3];

static bool	swap_dct = false;
		DWORD	contype;
		bool	day_changed,month_changed;

#ifdef _DEBUG
	connected = Fake_Connection();
	if(connected==TRUE)
	contype =  INTERNET_CONNECTION_MODEM;
	else
	contype = 0;
#else
	connected = InternetGetConnectedState(&contype, NULL);
	if(connected==TRUE)
	contype = (contype & INTERNET_CONNECTION_MODEM);
	else
	contype = 0;
#endif

	m_my_api.GetDay(&today,&day_changed,czprevi_day);
	month_changed= m_my_api.DetectMonChange();
	m_my_api.GetTime(m_now);

	//Do Rough calculation of duration (For eficiency)
	m_win_counter.Incriment(m_win_duration);

if(m_privilage==SERVER)
{//+++++++++++++++++++++++++++++++ (IF SERVER) ++++++++++++++++++++++++++++++++++
//DETECT CHANGE OF DAY
	if(day_changed)
	{
		if (contype==INTERNET_CONNECTION_MODEM)
		{//Save Internet log
			m_my_api.TimeElapse(m_int_str_sec,m_now,m_temp_duration);
			m_int_counter.GetRoughDuration(m_temp_rugh_dura);


if(m_my_api.Save_Int_Usage(&today,m_int_str_time,m_now,m_temp_duration,m_temp_rugh_dura)==false)

			{
				KillMeter();
				return;
			}
			memcpy(m_int_str_time,m_now,9);
			m_int_str_sec=m_my_api.ToSeconds(m_int_str_time);
			m_int_counter.Reset();
		}
		//Save Windows log
		m_my_api.TimeElapse(m_win_str_sec,m_now,m_temp_duration);
		m_win_counter.GetRoughDuration(m_temp_rugh_dura);


if(m_my_api.Save_Win_Usage(&today,m_win_str_time,m_now,m_temp_duration,m_temp_rugh_dura)==false)
```

```
                        {
                                KillMeter();
                                return;
                        }
                memcpy(m_win_str_time,m_now,9);
                m_win_str_sec=m_my_api.ToSeconds(m_win_str_time);
                m_win_counter.Reset();

                if (month_changed==false)
                {//If month has changed there is no need to load file because it loads in bellow
month change check
                //load log file
                        if(m_my_api.LoadLog()==false)
                        {//Cannot load the log file
                                KillMeter();
                                return;
                        }
                }
        }

        //DETECT CHANGE OF MONTH
        if (month_changed)
        {//Month shift
                m_my_api.MonthChange();
                //load log file
                if(m_my_api.LoadLog()==false)
                {//Cannot load the log file
                        KillMeter();
                        return;
                }
        }

//CHECK CONNECTION STATE
        if (contype==INTERNET_CONNECTION_MODEM)
        {
                if (swap_dct==false){
                swap_dct=true;
//-----------JUST CONNECTED----------------------
                memcpy(m_int_str_time,m_now,9);
                m_int_str_sec = m_my_api.ToSeconds(m_int_str_time);
                m_int_counter.Reset();
                SetTip(ID_MODE_CONECT,ID_TIP_PRAYORITY_HIGH);
                //Create "Connected" file to show connection state
                m_my_api.SetStatusFile(CONNECTED);
//-----------JUST CONNECTED----------------------
                }
        }else{
                if (swap_dct==true){
                swap_dct=false;
//---------JUST DISCONNECTED----------------------
                //Save Internet log
                m_my_api.TimeElapse(m_int_str_sec,m_now,m_temp_duration);
                m_int_counter.GetRoughDuration(m_temp_rugh_dura);


if(m_my_api.Save_Int_Usage(&today,m_int_str_time,m_now,m_temp_duration,m_temp_rugh_dura)==false)
                {
                        KillMeter();
                        return;
                }
                SetTip(ID_MODE_DESCONECT , ID_TIP_PRAYORITY_HIGH);
                //Create "Disconnected" file to show connection state
                m_my_api.SetStatusFile(DISCONNECTED);
//---------JUST DISCONNECTED----------------------
                }}

//GIVE DATA AND ANSWER TO CLIENT IF NESSASARY
        m_my_api.SayServerActive(m_win_str_time,m_win_counter.m_rough_sec);

}else{
//+++++++++++++++++++++++++++++++ (IF CLIENT) ++++++++++++++++++++++++++++++++++++++++

        //DETECT CHANGE OF DAY
```

```
	if(day_changed)
	{
		if (contype==INTERNET_CONNECTION_MODEM)
		{
		memcpy(m_int_str_time,m_now,9);
		m_int_str_sec=m_my_api.ToSeconds(m_int_str_time);
		m_int_counter.Reset();
		}

		memcpy(m_win_str_time,m_now,9);
		m_win_str_sec = m_my_api.ToSeconds(m_win_str_time);
		m_win_counter.Reset();

		if (month_changed==false)
		{//If month has changed there is no need to load file because it loads in bellow
month change check
			//Wait untill server setup the log file
			::Sleep(1100);
			if(m_my_api.LoadLog()==false)
			{
				KillMeter();
				return;
			}
		}
	}
	//Detect month changed
	if (month_changed)
	{
		m_my_api.MonthChange();
		//Wait untill server setup the log file
		::Sleep(1100);
		if(m_my_api.LoadLog()==false)
		{
			KillMeter();
			return;
		}
	}

//CHECK CONNECTION STATE
	if (contype==INTERNET_CONNECTION_MODEM)
	{
		if (swap_dct==false){
		swap_dct=true;
//-----------JUST CONNECTED----------------------
		memcpy(m_int_str_time,m_now,9);
		m_int_str_sec=m_my_api.ToSeconds(m_int_str_time);
		m_int_counter.Reset();
		SetTip(ID_MODE_CONECT,ID_TIP_PRAYORITY_HIGH);
//-----------JUST CONNECTED----------------------
		}
	}else{
		if (swap_dct==true){
		swap_dct=false;
//---------JUST DISCONNECTED----------------------
		//Wait untill server setup file
		::Sleep(1100);
		if(m_my_api.LoadLog()==false)
		{
			KillMeter();
			return;
		}
		SetTip(ID_MODE_DESCONECT,ID_TIP_PRAYORITY_HIGH);
//---------JUST DISCONNECTED----------------------
		}}
}
//++++++++++++++++++++++ (COMMON TO SERVER ANFD CLIENT) ++++++++++++++++++++++++++

//Setup Tip
	if (swap_dct==true){
	//Connected
	//Do Rough calculation of duration (For eficiency)
	m_int_counter.Incriment(m_int_duration);
	SetTip(ID_MODE_CONECT , ID_TIP_PRAYORITY_NORMAL);
```

```
	}else{
	//Disconnected
	SetTip(ID_MODE_DESCONECT , ID_TIP_PRAYORITY_NORMAL);
	}
//Setup label
	if (!IsWindowVisible())
		return ;//Execut only if visible
	if (swap_dct==true){
	//Connected
	SetLable(ID_MODE_CONECT);
	}else{
	//Disconnected
	SetLable(ID_MODE_DESCONECT);
	}
}
//-----------------------------------------------------------------------------------------
void CMainDlg::OnFileViewer()
{
char   path[_MAX_PATH];
DWORD  len = _MAX_PATH;
DWORD  last_slash;
STARTUPINFO si;
PROCESS_INFORMATION pi;

	//get app path
	GetAppPath(path,&len,false);
	//append "\\Viewer.exe" to the path
	last_slash = FindLatsSlash(path,len);
	memcpy(path+last_slash,"\\Viewer.exe\0",12);

	ZeroMemory(&si,sizeof(STARTUPINFO));
	//memset(&si,0,sizeof(STARTUPINFO));
	si.wShowWindow=1;
	si.cb=sizeof(STARTUPINFO);
	si.dwFlags=STARTF_USESTDHANDLES;

	if(CreateProcess(path,NULL,NULL,NULL,FALSE,NORMAL_PRIORITY_CLASS,NULL,NULL,&si,&pi)==0)
		::MessageBox(this->m_hWnd,"File Viewer cannot be load !    ","Usage Meter",MB_OK|
MB_ICONEXCLAMATION);
}
//-----------------------------------------------------------------------------------------
void CMainDlg::GetAppPath(char* path,DWORD* len,bool writerunkey)
{
	//Get app path
	*len = GetModuleFileName(NULL,path,*len);

#ifdef _DEBUG
#else
	if(writerunkey==true)
	{//write Meter path to Run key
	SECURITY_ATTRIBUTES sa;
	HKEY hKey;
		sa.nLength=sizeof(sa);
		sa.bInheritHandle=1;
		sa.lpSecurityDescriptor=NULL;
		//Open or Create Key


RegCreateKeyEx(HKEY_LOCAL_MACHINE,"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",NULL,NULL,
REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS, &sa, &hKey,NULL);
		//Set Value
		RegSetValueEx(hKey,"UsageMeter",0,REG_SZ,(LPBYTE)path, *len);
		//Close
		RegCloseKey(hKey);
	}
#endif
}
//-----------------------------------------------------------------------------------------
DWORD CMainDlg::FindLatsSlash(char *path,DWORD len)
{
	for(DWORD i=len;i>0;i--)
	{
		if(path[i]=='\\')
```

```
                    return i;
        }
        return 0;
}

// CAboutDlg dialog used for App
About******************************************************
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
        ON_WM_CLOSE()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

//-----------------------------------------------------------------------------------------
void CAboutDlg::OnClose()
{//Set pointer to CAboutDlg handle to NULL
        *m_my_handle=NULL;
        CDialog::OnClose();
}
//-----------------------------------------------------------------------------------------
void CAboutDlg::OnOK()
{//Set pointer to CAboutDlg handle to NULL
        *m_my_handle=NULL;
        CDialog::OnOK();
}
//-----------------------------------------------------------------------------------------

BOOL CAboutDlg::OnInitDialog()
{//Give CAboutDlg handle to CMainDlg
        *m_my_handle=this;
        return TRUE;
}
//-----------------------------------------------------------------------------------------

void CMainDlg::OnBeepSet()
{
CDlgBeepSetting     m_dlg_beep_set;
        m_dlg_beep_set.DoModal();
        SetBeep(m_dlg_beep_set.m_reset_int_count, m_dlg_beep_set.m_reset_win_count);
}
//-----------------------------------------------------------------------------------------
void CMainDlg::SetBeep(bool reset_int_count, bool reset_win_count)
{
BEEP_STRUCT         ps;
        //Internet
        m_int_counter.RetriveSettings(&ps,false);
        m_int_counter.CheckInputs(&ps,false,NULL);
        m_int_counter.SetBeep(&ps,reset_int_count);
        //Windows
        m_win_counter.RetriveSettings(&ps,true);
        m_win_counter.CheckInputs(&ps,false,NULL);
        m_win_counter.SetBeep(&ps,reset_win_count);
}
//-----------------------------------------------------------------------------------------
void CMainDlg::KillMeter()
{
        ::MessageBox(this->m_hWnd,"Cannot open the Log File, Usage Meter is terminating!
","Usage Meter",MB_OK|MB_ICONSTOP);
        delete m_pop_menu;
        KillTimer(TIMER_SERVER_LINK);
        KillTimer(TIMER_MAIN);
        SetUpTray(NIM_DELETE,NULL,0);
```

```
	SaveWindowPos();
	m_my_api.RemoveUser();
	CDialog::EndDialog(0);
}
//-----------------------------------------------------------------------------
#ifdef _DEBUG
void CMainDlg::OnLButtonUp(UINT nFlags, CPoint point)
{
	if(m_privilage==SERVER)
		AfxMessageBox("SERVER");
	else
		AfxMessageBox("CLIENT");
	CDialog::OnLButtonUp(nFlags, point);
}
//-----------------------------------------------------------------------------
bool CMainDlg::Fake_Connection()
{
DWORD i;
	i=m_my_api.RegGet(HKEY_LOCAL_MACHINE,"Connected");
	if(i==1)
		return true;
	return false;
}
#endif

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// Usage Meter.h : main header file for the USAGE METER application
//

#if !defined(AFX_USAGEMETER_H__A78CEC0C_5E17_47F5_BDC6_3F47D12423AD__INCLUDED_)
#define AFX_USAGEMETER_H__A78CEC0C_5E17_47F5_BDC6_3F47D12423AD__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"		// main symbols
#include "MainDlg.h"	// Added by ClassView

/////////////////////////////////////////////////////////////////////////////
// CUsageMeterApp:
// See Usage Meter.cpp for the implementation of this class
//
class CUsageMeterApp : public CWinApp
{
public:
	CUsageMeterApp();

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CUsageMeterApp)
	public:
	virtual BOOL InitInstance();
	virtual BOOL PreTranslateMessage(MSG* pMsg);
	//}}AFX_VIRTUAL

// Implementation

	//{{AFX_MSG(CUsageMeterApp)
		// NOTE - the ClassWizard will add and remove member functions here.
		//    DO NOT EDIT what you see in these blocks of generated code !
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
protected:
	BOOL HasPreInstance();
};
/////////////////////////////////////////////////////////////////////////////
```

```
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.

#endif // !defined(AFX_USAGEMETER_H__A78CEC0C_5E17_47F5_BDC6_3F47D12423AD__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// Usage Meter.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "Usage Meter.h"
#include "MyLogFile.h"
#include "MainDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CUsageMeterApp
BEGIN_MESSAGE_MAP(CUsageMeterApp, CWinApp)
	//{{AFX_MSG_MAP(CUsageMeterApp)
		// NOTE - the ClassWizard will add and remove mapping macros here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG
	ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CUsageMeterApp construction

CUsageMeterApp::CUsageMeterApp()
{
	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CUsageMeterApp object
CUsageMeterApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CUsageMeterApp initialization

BOOL CUsageMeterApp::InitInstance()
{
#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC in a shared DLL
#else
	Enable3dControlsStatic();	// Call this when linking to MFC statically
#endif

	if(HasPreInstance()==FALSE)
	{
	CMainDlg dlg ;
	m_pMainWnd =&dlg;
	dlg.DoModal();
	}
	return FALSE;
}

BOOL CUsageMeterApp::PreTranslateMessage(MSG* pMsg)
{
#ifdef _DEBUG
#else
	// Detect Esc KeyPrees and bypass it
	if(pMsg->message==WM_KEYDOWN)
	{
		if(pMsg->wParam==27)
```

```
                {ShowWindow(theApp.m_pMainWnd->m_hWnd,0);
                return 0;}
        }
#endif
return CWinApp::PreTranslateMessage(pMsg);
}

BOOL CUsageMeterApp::HasPreInstance()
{
#ifdef _DEBUG
        return FALSE;
#else
CMyLogFile  lf;
CWnd*         prewnd = CWnd::FindWindow("#32770", lf.CreateTitle());
        if(prewnd)
        {
                prewnd->ShowWindow(1);
                prewnd->SetForegroundWindow();
                return TRUE;
        }else{
                return FALSE;
        }
#endif
}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

#if !defined(AFX_DLGPATH_H__4B2F9AA0_2912_459C_A49F_05D2E6871AA1__INCLUDED_)
#define AFX_DLGPATH_H__4B2F9AA0_2912_459C_A49F_05D2E6871AA1__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// DlgPath.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CDlgPath dialog

#include  "MyRegistry.h"

class CDlgPath : public CDialog
{
// Construction
public:
        CDlgPath(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
        //{{AFX_DATA(CDlgPath)
        enum { IDD = IDD_DIALOG_PATH };
        CEdit  m_txt_path;
        //}}AFX_DATA


// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CDlgPath)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:

        // Generated message map functions
        //{{AFX_MSG(CDlgPath)
        virtual void OnOK();
        virtual void OnCancel();
        virtual BOOL OnInitDialog();
        afx_msg void OnTimer(UINT nIDEvent);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
private:
```

```
	void MySleep();
	CMyRegistry m_clsreg;
	bool		m_bypass;
public:
	DWORD*		m_pprivilage;
	CString*	m_plogpath;
	bool		m_errer;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.

#endif // !defined(AFX_DLGPATH_H__4B2F9AA0_2912_459C_A49F_05D2E6871AA1__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// DlgPath.cpp : implementation file
//

#include "stdafx.h"
#include "usage meter.h"
#include "DlgPath.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
CDlgPath::CDlgPath(CWnd* pParent /*=NULL*/)
	: CDialog(CDlgPath::IDD, pParent)
{
	//{{AFX_DATA_INIT(CDlgPath)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
}
void CDlgPath::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CDlgPath)
	DDX_Control(pDX, IDC_EDIT_PATH, m_txt_path);
	//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CDlgPath, CDialog)
	//{{AFX_MSG_MAP(CDlgPath)
	ON_WM_TIMER()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
void CDlgPath::OnOK()
{
WIN32_FIND_DATA fd;
char   path[255];
LPTSTR msg;
CString	temp;
HANDLE hfind;

	//Get window text
	int i = m_txt_path.GetWindowText(path,255);

	if(i==0)
	{//There is no text in the texbox
		::MessageBox(NULL,"Please enter the path!   ","Usage Meter",MB_OK|
MB_ICONEXCLAMATION);
		return;
	}
	//Add '*' this is needed
	if(path[i-1]=='\\')
		memcpy(path+i,"*\0",2);
	else
		memcpy(path+i,"\\*\0",3);
```

```
	hfind = FindFirstFile(path, &fd);
	if((hfind==INVALID_HANDLE_VALUE)||(path[1]!=':'))
	{//Folder creation fails
		FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER|
FORMAT_MESSAGE_FROM_SYSTEM,NULL,GetLastError(),MAKELANGID(LANG_ENGLISH,
SUBLANG_ENGLISH_US),(LPTSTR)&msg,0,NULL);
		temp.Format("%s%s","Unable to locate specified folder, Please check the path !
\r\n System Error - ",msg);
		::MessageBox(NULL,temp,"Usage Meter",MB_OK|MB_ICONEXCLAMATION);
		LocalFree(msg);
	}else{
		FindClose(hfind);
		KillTimer(TIMER_GET_PATH);
		m_bypass = true;
		m_errer = false;
		*m_plogpath = path;
		m_clsreg.RegSet(HKEY_LOCAL_MACHINE,"LogFolder",m_plogpath->GetBuffer(m_plogpath-
>GetLength()),m_plogpath->GetLength());
		m_clsreg.RegSet(HKEY_LOCAL_MACHINE,"PathPort",MSG_GOT_PATH);
		MySleep();
		CDialog::OnOK();
	}
}
//-----------------------------------------------------------------------------------
void CDlgPath::OnCancel()
{
	KillTimer(TIMER_GET_PATH);
	m_bypass = true;
	m_clsreg.RegSet(HKEY_LOCAL_MACHINE,"PathPort",MSG_PATH_ERROR);
	m_errer = true;
	CDialog::OnCancel();
}
//-----------------------------------------------------------------------------------


BOOL CDlgPath::OnInitDialog()
{
	CDialog::OnInitDialog();
	m_clsreg.RegSet(HKEY_LOCAL_MACHINE,"PathPort",MSG_INVALID);
	SetTimer(TIMER_GET_PATH,500,0);
	return TRUE;
}
//-----------------------------------------------------------------------------------
void CDlgPath::OnTimer(UINT nIDEvent)
{
	if(m_bypass!=true)
	{
		if(TIMER_GET_PATH==nIDEvent)
		{
			if(m_clsreg.RegGet(HKEY_LOCAL_MACHINE,"PathPort")==MSG_GOT_PATH)
			{//Some one has setup the path
				KillTimer(TIMER_GET_PATH);
				m_errer = false;
				//sleep a while if client becauose server allway update the file
				MySleep();
				char   czusers[100];
				DWORD  size = 100;
				m_clsreg.RegGet(HKEY_LOCAL_MACHINE,"LogFolder",czusers,&size);
				*m_plogpath = czusers;
				CDialog::EndDialog(0);
			}
			if(m_clsreg.RegGet(HKEY_LOCAL_MACHINE,"PathPort")==MSG_PATH_ERROR)
			{
				KillTimer(TIMER_GET_PATH);
				m_errer = true;
				CDialog::EndDialog(0);
			}
		}
	}

	CDialog::OnTimer(nIDEvent);
}
//-----------------------------------------------------------------------------------
```

```
void CDlgPath::MySleep()
{
	if(*m_pprivilage==CLIENT)
	{
		this->ShowWindow(0);
		::Sleep(1100);
	}
}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

#if !defined(AFX_DLGBEEPSETTING_H__9155CB2E_FA25_4A38_B90A_65F7738AC5AB__INCLUDED_)
#define AFX_DLGBEEPSETTING_H__9155CB2E_FA25_4A38_B90A_65F7738AC5AB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// DlgBeepSetting.h : header file
//////////////////////////////////////////////////////////////////////////
// CDlgBeepSetting dialog

#include "MyBeep.h"

class CDlgBeepSetting : public CDialog
{
// Construction
public:
	CDlgBeepSetting(CWnd* pParent = NULL);   // standard constructor
// Dialog Data
	//{{AFX_DATA(CDlgBeepSetting)
	enum { IDD = IDD_BEEP_DLG };
	CEdit  m_win_inter;
	CEdit  m_win_free;
	CEdit  m_win_dur;
	CEdit  m_int_inter;
	CEdit  m_int_free;
	CEdit  m_int_dur;
	CButton	m_win_chk;
	CButton	m_int_chk;
	CStatic	m_fram;
	CTabCtrl	m_Tab_Beep;
	//}}AFX_DATA
// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CDlgBeepSetting)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL
// Implementation
protected:
	// Generated message map functions
	//{{AFX_MSG(CDlgBeepSetting)
	afx_msg void OnTestBeep();
	virtual void OnOK();
	virtual BOOL OnInitDialog();
	afx_msg void OnSelchangingTabBeep(NMHDR* pNMHDR, LRESULT* pResult);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
public:
	bool m_reset_int_count, m_reset_win_count;
private:
	CMyBeep	m_beep_cls;
	int		m_old_int_interval, m_old_win_interval;
	int		m_old_int_chkstate, m_old_win_chkstate;
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.
#endif // !defined(AFX_DLGBEEPSETTING_H__9155CB2E_FA25_4A38_B90A_65F7738AC5AB__INCLUDED_)
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// DlgBeepSetting.cpp : implementation file
//
```

```
#include "stdafx.h"
#include "usage meter.h"
#include "DlgBeepSetting.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CDlgBeepSetting dialog
CDlgBeepSetting::CDlgBeepSetting(CWnd* pParent /*=NULL*/)
	: CDialog(CDlgBeepSetting::IDD, pParent)
{
	//{{AFX_DATA_INIT(CDlgBeepSetting)
	//}}AFX_DATA_INIT
}

void CDlgBeepSetting::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CDlgBeepSetting)
	DDX_Control(pDX, EDIT_WIN_INTER, m_win_inter);
	DDX_Control(pDX, EDIT_WIN_FREE, m_win_free);
	DDX_Control(pDX, EDIT_WIN__DUR, m_win_dur);
	DDX_Control(pDX, EDIT_INT_INTER, m_int_inter);
	DDX_Control(pDX, EDIT_INT_FREE, m_int_free);
	DDX_Control(pDX, EDIT_INT__DUR, m_int_dur);
	DDX_Control(pDX, CHK_WIN_BEEP, m_win_chk);
	DDX_Control(pDX, CHK_INT_BEEP, m_int_chk);
	DDX_Control(pDX, IDC_STATIC_FRAM, m_fram);
	DDX_Control(pDX, IDC_TAB_BEEP, m_Tab_Beep);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CDlgBeepSetting, CDialog)
	//{{AFX_MSG_MAP(CDlgBeepSetting)
	ON_BN_CLICKED(IDC_TEST_BEEP, OnTestBeep)
	ON_NOTIFY(TCN_SELCHANGING, IDC_TAB_BEEP, OnSelchangingTabBeep)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CDlgBeepSetting message handlers
BOOL CDlgBeepSetting::OnInitDialog()
{
	CDialog::OnInitDialog();
	//Arrangeg coltrols
	m_Tab_Beep.InsertItem(0,"Internet");
	m_Tab_Beep.InsertItem(1,"Windows");
	m_Tab_Beep.SetWindowPos(&CWnd::wndBottom,5,4,235,94,SWP_SHOWWINDOW);
	m_fram.SetWindowPos(&CWnd::wndBottom,9,28,226,65,SWP_SHOWWINDOW);
RECT lpRect;
	this->GetWindowRect(&lpRect);
	this-
>SetWindowPos(&CWnd::wndBottom,lpRect.left+200,lpRect.top+308,249,149,SWP_SHOWWINDOW);

	//Internet tab selected first
	m_win_free.ShowWindow(0);
	m_win_dur.ShowWindow(0);
	m_win_inter.ShowWindow(0);
	m_win_chk.ShowWindow(0);
	m_int_free.SetWindowPos(&CWnd::wndBottom,73,37,35,20,SWP_SHOWWINDOW);
	m_int_inter.SetWindowPos(&CWnd::wndBottom,182,37,23,20,SWP_SHOWWINDOW);
	m_int_dur.SetWindowPos(&CWnd::wndBottom,73,65,35,20,SWP_SHOWWINDOW);
	m_int_chk.SetWindowPos(&CWnd::wndBottom,146,67,77,20,SWP_SHOWWINDOW);

	//Re_stor controls values
BEEP_STRUCT bs;
char   temp[5];
	//Internet
```

```
	m_beep_cls.RetriveSettings(&bs,false);
	m_beep_cls.CheckInputs(&bs,false,NULL);
	m_old_int_interval = bs.Interval;
	m_int_free.SetWindowText(_ltoa(bs.Frequency,temp,10));
	m_int_dur.SetWindowText(_ltoa(bs.Duration,temp,10));
	m_int_inter.SetWindowText(_ltoa(bs.Interval,temp,10));
	if(bs.Enable==1)
	{
		m_old_int_chkstate = 1;
		m_int_chk.SetCheck(1);
	}else{
		m_old_int_chkstate = 0;
		m_int_chk.SetCheck(0);
	}
	//Windows
	m_beep_cls.RetriveSettings(&bs,true);
	m_beep_cls.CheckInputs(&bs,false,NULL);
	m_old_win_interval = bs.Interval;
	//Re-stor controls values
	m_win_free.SetWindowText(_ltoa(bs.Frequency,temp,10));
	m_win_dur.SetWindowText(_ltoa(bs.Duration,temp,10));
	m_win_inter.SetWindowText(_ltoa(bs.Interval,temp,10));
	if(bs.Enable==1)
	{
		m_old_win_chkstate = 1;
		m_win_chk.SetCheck(1);
	}else{
		m_old_win_chkstate = 0;
		m_win_chk.SetCheck(0);
	}
	return TRUE;
}
//-----------------------------------------------------------------------------
void CDlgBeepSetting::OnTestBeep()
{
BEEP_STRUCT bs;
char		temp[5];
	//Set fake value to Interval member
	bs.Interval = 10;

	if(m_Tab_Beep.GetItemState(1,1) == 0)
	{//Internet
		m_int_free.GetWindowText(temp,5);
		bs.Frequency = atoi(temp);
		m_int_dur.GetWindowText(temp,5);
		bs.Duration = atoi(temp);
	}
	else
	{//Windows
		m_win_free.GetWindowText(temp,5);
		bs.Frequency = atoi(temp);
		m_win_dur.GetWindowText(temp,5);
		bs.Duration = atoi(temp);
	};
	//Do beep
	if(m_beep_cls.CheckInputs(&bs,true,this->m_hWnd)==true)
		Beep(bs.Frequency,bs.Duration);
}
//-----------------------------------------------------------------------------
void CDlgBeepSetting::OnOK()
{
BEEP_STRUCT bs;
char		temp[5];
int			new_chk_state;
//Internet
*******************************************************************************
	m_int_free.GetWindowText(temp,5);
	bs.Frequency = atoi(temp);
	m_int_dur.GetWindowText(temp,5);
	bs.Duration = atoi(temp);
	m_int_inter.GetWindowText(temp,5);
	bs.Interval = atoi(temp);
```

```
	if(m_beep_cls.CheckInputs(&bs,true,this->m_hWnd)==true)
	{
		m_beep_cls.RegSet(HKEY_CURRENT_USER,"IntFrequency",(DWORD)bs.Frequency);
		m_beep_cls.RegSet(HKEY_CURRENT_USER,"IntDuration",(DWORD)bs.Duration);
		m_beep_cls.RegSet(HKEY_CURRENT_USER,"IntInterval",(DWORD)bs.Interval);
		if(m_int_chk.GetCheck()==1)
		{
			new_chk_state = 1;
			m_beep_cls.RegSet(HKEY_CURRENT_USER,"IntEnable",(DWORD)1);
		}
		else
		{
			new_chk_state = 0;
			m_beep_cls.RegSet(HKEY_CURRENT_USER,"IntEnable",(DWORD)0);
		}
	//If interval dosn't changed there is no neet to change
		//the counters beep count (m_reset_count's value check in Main Dialog)
	if((m_old_int_interval==bs.Interval)&&(m_old_int_chkstate==new_chk_state))
		m_reset_int_count = false;
	else
		m_reset_int_count = true;
	}
	else
		return;
//Windows
*********************************************************************************

	m_win_free.GetWindowText(temp,5);
	bs.Frequency = atoi(temp);
	m_win_dur.GetWindowText(temp,5);
	bs.Duration = atoi(temp);
	m_win_inter.GetWindowText(temp,5);
	bs.Interval = atoi(temp);

	if(m_beep_cls.CheckInputs(&bs,true,this->m_hWnd)==true)
	{
		m_beep_cls.RegSet(HKEY_CURRENT_USER,"WinFrequency",(DWORD)bs.Frequency);
		m_beep_cls.RegSet(HKEY_CURRENT_USER,"WinDuration",(DWORD)bs.Duration);
		m_beep_cls.RegSet(HKEY_CURRENT_USER,"WinInterval",(DWORD)bs.Interval);
		if(m_win_chk.GetCheck()==1)
		{
			new_chk_state = 1;
			m_beep_cls.RegSet(HKEY_CURRENT_USER,"WinEnable",(DWORD)1);
		}
		else
		{
			new_chk_state = 0;
			m_beep_cls.RegSet(HKEY_CURRENT_USER,"WinEnable",(DWORD)0);
		}

		if((m_old_win_interval==bs.Interval)&&(m_old_win_chkstate==new_chk_state))
		m_reset_win_count = false;
		else
		m_reset_win_count = true;
	CDialog::OnOK();
	}
}
//-------------------------------------------------------------------------------
void CDlgBeepSetting::OnSelchangingTabBeep(NMHDR* pNMHDR, LRESULT* pResult)
{
//Tab changed
		if(m_Tab_Beep.GetItemState(1,1) == 1)
		{//Internet
			m_win_free.ShowWindow(0);
			m_win_dur.ShowWindow(0);
			m_win_inter.ShowWindow(0);
			m_win_chk.ShowWindow(0);
			m_int_free.SetWindowPos(&CWnd::wndBottom,73,37,35,20,SWP_SHOWWINDOW);
			m_int_inter.SetWindowPos(&CWnd::wndBottom,182,37,23,20,SWP_SHOWWINDOW);
			m_int_dur.SetWindowPos(&CWnd::wndBottom,73,65,35,20,SWP_SHOWWINDOW);
			m_int_chk.SetWindowPos(&CWnd::wndBottom,146,67,77,20,SWP_SHOWWINDOW);
		}
		else
```

```
		{//Windows
			m_int_free.ShowWindow(0);
			m_int_dur.ShowWindow(0);
			m_int_inter.ShowWindow(0);
			m_int_chk.ShowWindow(0);
			m_win_free.SetWindowPos(&CWnd::wndBottom,73,37,35,20,SWP_SHOWWINDOW);
			m_win_inter.SetWindowPos(&CWnd::wndBottom,182,37,23,20,SWP_SHOWWINDOW);
			m_win_dur.SetWindowPos(&CWnd::wndBottom,73,65,35,20,SWP_SHOWWINDOW);
			m_win_chk.SetWindowPos(&CWnd::wndBottom,146,67,77,20,SWP_SHOWWINDOW);
		};
	*pResult = 0;
}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyBeep.h: interface for the CMyBeep class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_MYBEEP_H__06577F30_32C0_41B2_874D_2E8374E5C471__INCLUDED_)
#define AFX_MYBEEP_H__06577F30_32C0_41B2_874D_2E8374E5C471__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "MyTimeMath.h"

struct BEEP_STRUCT
{
	bool Enable;
	int    Duration;
	int Frequency;
	int Interval;
	int Count;
};

class CMyBeep : public CMyTimeMath
{
public:
	void ResetBeepCount();
	CMyBeep();
	virtual ~CMyBeep();

void Beep();
void SetBeep(BEEP_STRUCT* bs,bool reset_count);
void RetriveSettings(BEEP_STRUCT* bs, bool flag);
bool CheckInputs(BEEP_STRUCT* bs, bool show_err,HWND parent);
private:
	BEEP_STRUCT m_beep_info;
};

#endif // !defined(AFX_MYBEEP_H__06577F30_32C0_41B2_874D_2E8374E5C471__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyBeep.cpp: implementation of the CMyBeep class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "usage meter.h"
#include "MyBeep.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
//////////////////////////////////////////////////////////////////////
CMyBeep::CMyBeep(){}
//--------------------------------------------------------------------------
CMyBeep::~CMyBeep(){}
```

```
//-----------------------------------------------------------------------------
void CMyBeep::Beep()
{//Call by Counter Incrimet event (Client and Server both try to
	//beep but XP only execute active users ask
	if(m_beep_info.Enable==true)
	{
		m_beep_info.Count++;
		if(m_beep_info.Count==m_beep_info.Interval)
		{
			::Beep(m_beep_info.Frequency,m_beep_info.Duration);
			ResetBeepCount();
		}
	}
}
//-----------------------------------------------------------------------------
void CMyBeep::SetBeep(BEEP_STRUCT* bs,bool reset_count)
{
	m_beep_info.Enable		= bs->Enable;
	m_beep_info.Duration		= bs->Duration;
	m_beep_info.Frequency		= bs->Frequency;
	m_beep_info.Interval		= (bs->Interval*60);
	if(reset_count==true)
		ResetBeepCount();
}
//-----------------------------------------------------------------------------
void CMyBeep::RetriveSettings(BEEP_STRUCT* bs, bool flag)
{
	if(flag==false)
	{//Internet
		bs->Frequency= RegGet(HKEY_CURRENT_USER,"IntFrequency");
		bs->Duration = RegGet(HKEY_CURRENT_USER,"IntDuration");
		bs->Interval = RegGet(HKEY_CURRENT_USER,"IntInterval");
		if(RegGet(HKEY_CURRENT_USER,"IntEnable")==1)
			bs->Enable	= 1;
		else
			bs->Enable	= 0;
	}else{
	//Windows
		bs->Frequency= RegGet(HKEY_CURRENT_USER,"WinFrequency");
		bs->Duration = RegGet(HKEY_CURRENT_USER,"WinDuration");
		bs->Interval = RegGet(HKEY_CURRENT_USER,"WinInterval");
		if(RegGet(HKEY_CURRENT_USER,"WinEnable")==1)
			bs->Enable	= 1;
		else
			bs->Enable	= 0;
	}
}
//-----------------------------------------------------------------------------
bool CMyBeep::CheckInputs(BEEP_STRUCT* bs, bool show_err,HWND parent)
{
CString	msg;
bool	err = false;

	if((bs->Frequency<1000)||(bs->Frequency>4000))
	{
		err = true;
		bs->Frequency = 2500;
		msg = "Frequency must be in between 1000Hz - 4000Hz ";
	}
	if((bs->Duration<10)||(bs->Duration>2000))
	{
		err = true;
		bs->Duration = 100;
		msg = "Duration must be in between 10ms - 2000ms ";
	}
	if((bs->Interval<1)||(bs->Interval>60))
	{
		err = true;
		bs->Interval = 10;
		msg = "Interval must be in between 1min - 60min ";
	}

	if(err==true)
```

```
	{
		if(show_err==true)
		{
		::MessageBox(NULL,msg,"Usage Meter",MB_OK|MB_ICONEXCLAMATION);
		}
		return false;
	}else{
		return true;
	}
}
//--------------------------------------------------------------------------
void CMyBeep::ResetBeepCount()
{
	m_beep_info.Count = 0;
}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyLogFile.h: interface for the CMyLogFile class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_MYLOGFILE_H__931E4F8F_5F7B_4067_92CB_1DCB82863187__INCLUDED_)
#define AFX_MYLOGFILE_H__931E4F8F_5F7B_4067_92CB_1DCB82863187__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "MyLogFolder.h"

struct IU_FILE_STRUCT
{
char   type;
char   day[3];
char   start_end[22];
char   duration[9];
char   roughdu[9];
DWORD  names_len;
};

class CMyLogFile : public CMyLogFolder
{
public:
	bool   Save_Int_Usage(DAY_STRUCT* day,char* start,char* end,char* duration,char* roughdu);
	bool   Save_Win_Usage(DAY_STRUCT* day,char* start,char* end,char* duration,char* roughdu);
	void   IntLogFileObj(HWND hint_list,HWND hwin_list,char* pint_total,char* pwin_total,CWnd*
pmaindlg,DWORD* pprivilage);
	void   SetStatusFile(WORD flag);
	CString      CreateTitle();
	void   MonthChange();
	bool   LoadLog();

private:
	void FormatTotal(char *buf, DWORD seconds);
	bool SaveLog(char* names);

	CWinThread* m_pmythread;
	CString             m_logfilename;
	HWND          m_hint_list, m_hwin_list;
	char          *m_pint_total,*m_pwin_total;
	CWnd*         m_pmaindlg;
IU_FILE_STRUCT      m_us;
};

#endif // !defined(AFX_MYLOGFILE_H__931E4F8F_5F7B_4067_92CB_1DCB82863187__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyLogFile.cpp: implementation of the CMyLogFile class.
//
//////////////////////////////////////////////////////////////////////
```

```
#include "stdafx.h"
#include "Usage Meter.h"
#include "MyLogFile.h"
#include "MyListView.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

#define      IU_FILE_SRT_SIZE    48
#define TYPE_WINDOWS             0
#define TYPE_INTERNET            1

//////////////////////////////////////////////////////////////////////////////////////
void CMyLogFile::IntLogFileObj(HWND hint_list,HWND hwin_list,char* pint_total,char*
pwin_total,CWnd* pmaindlg,DWORD* pprivilage)
{

       m_us.day[2]               = '\0';
       m_us.start_end[21]  = '\0';
       m_us.duration[8]    = '\0';
       m_us.roughdu[8]           = '\0';

       m_pint_total= pint_total;
       m_pwin_total= pwin_total;
       m_hint_list   = hint_list;
       m_hwin_list   = hwin_list;
       m_pmaindlg    = pmaindlg;
       CMyLogFolder::m_pprivilage = pprivilage;
       m_pmythread   = AfxGetApp();
       MonthChange();
}
//////////////////////////////////////////////////////////////////////////////////////
bool CMyLogFile::LoadLog()
{
CFile  LF;
DWORD  filelenth;
char*  filedata = NULL;
char   buffer[255];
CString logpath;
CMyListView         list;
CFileException      ex;

       m_pmythread->SetThreadPriority(THREAD_PRIORITY_HIGHEST);

       //Get loging path
       if(GetLogFolder(&logpath)==false)
              return false;
       else//m_logfilename set from IntLogFileObj by MonthChange
              logpath = logpath + m_logfilename;

       if(LF.Open(logpath,CFile::modeRead)!=1)
       {//File can't open so create it
              if(LF.Open(logpath,CFile::modeRead|CFile::modeCreate,&ex)!=1)
              {
                     ex.GetErrorMessage(buffer,255);
                     ::MessageBox(m_pmaindlg->m_hWnd,buffer,"Usage Meter",MB_OK|MB_ICONSTOP);
                     return false;//Still error occurs lets stop this
              }
       }
       filelenth = LF.GetLength();
       if(filelenth==0)
       {//file has no data
              list.DeleteAll(m_hwin_list);
              list.DeleteAll(m_hint_list);
              m_pmythread->SetThreadPriority(THREAD_PRIORITY_NORMAL);
              return true;
       }
       filedata = new char[filelenth];
       LF.Read(filedata,filelenth);
       LF.Close();
```

```
//********* Fill Lists ************************************************
int                 win_total_sec=0;
int                 int_total_sec=0;
int                 win_ruf_total_sec=0;
int                 int_ruf_total_sec=0;
int                 int_index=0;
int                 win_index=0;
int                 *ptotal,*pruftotal,*pindex;
DWORD         p=0;
char          log_no[3];

       list.DrawList(m_hwin_list,FALSE);
       list.DrawList(m_hint_list,FALSE);
       list.DeleteAll(m_hwin_list);
       list.DeleteAll(m_hint_list);

do{
       //Fill structure
       memcpy(&m_us,filedata+p,IU_FILE_SRT_SIZE);

       //Get names******************
       p = p+IU_FILE_SRT_SIZE;
       buffer[m_us.names_len] = '\0';
       memcpy(buffer,filedata+p,m_us.names_len);
       p = p+m_us.names_len;
       //Get names******************

       //Filter records
       if(m_us.type == TYPE_INTERNET){
       list.m_hlistview = m_hint_list;
       int_index++;
       pindex=&int_index;
       ptotal=&int_total_sec;
       pruftotal=&int_ruf_total_sec;
       NumToString(log_no,int_index);
       }else{
       list.m_hlistview = m_hwin_list;
       win_index++;
       pindex=&win_index;
       ptotal=&win_total_sec;
       pruftotal=&win_ruf_total_sec;
       NumToString(log_no,win_index);}

       //Insert log
       list.InsertRow(*pindex,log_no,&m_us,buffer);

       //Calcutate total
       TimeAddition(ptotal,m_us.duration);
       //Calcutate rough total
       TimeAddition(pruftotal,m_us.roughdu);

}while(p<filelenth);

       delete[] filedata;
       list.DrawList(m_hwin_list,TRUE);
       list.DrawList(m_hint_list,TRUE);
//********* Fill Lists ************************************************

       //set Totals
       FormatTotal(m_pwin_total,win_total_sec);
       FormatTotal(m_pint_total,int_total_sec);

       //set rough totals**************
       filedata = &buffer[1];
       FormatTotal(filedata+1,win_ruf_total_sec);
       filedata[0] = '<';filedata[10] = '>';filedata[11] = '\0';
       list.InsertTotal(win_index,m_hwin_list,filedata);

       FormatTotal(filedata+1,int_ruf_total_sec);
       list.InsertTotal(int_index,m_hint_list,filedata);
       //set rough totals**************
```

```
	m_pmythread->SetThreadPriority(THREAD_PRIORITY_NORMAL);
	return true;
}
//-----------------------------------------------------------------------------------
bool CMyLogFile::Save_Int_Usage(DAY_STRUCT* day,char* start,char* end,char* duration,char*
roughdu)
{//Save internet usage to the Log file

	m_us.type = TYPE_INTERNET;
	memcpy(m_us.day,day->day,2);

char st_en[22];
	memcpy(st_en,start,8);
	memcpy(st_en+8,"  -  ",5);
	memcpy(st_en+13,end,9);
	memcpy(m_us.start_end,st_en,22);

	memcpy(m_us.duration, duration,9);
	memcpy(m_us.roughdu, roughdu,9);
	//Get user list and lenth
char   users[200];
DWORD  size = 200;
		GetIntUsers(users,&size);
		m_us.names_len = size;
bool   ret = SaveLog(users);

	if(ret==true)
	{//reload log file
		LoadLog();
		return true;
	}else
		return false;
}
//-----------------------------------------------------------------------------------
bool CMyLogFile::Save_Win_Usage(DAY_STRUCT* day,char* start,char* end,char* duration,char*
roughdu)
{//Save windows usage to the Log file

	m_us.type = TYPE_WINDOWS;
	memcpy(m_us.day, day, 1);

char st_en[22];
	memcpy(st_en,start,8);
	memcpy(st_en+8,"  -  ",5);
	memcpy(st_en+13,end,9);
	memcpy(m_us.start_end,st_en,22);

	memcpy(m_us.duration, duration,9);
	memcpy(m_us.roughdu, roughdu,9);

char users[100];
	m_us.names_len = 100;
	GetWinUsers(users,&m_us.names_len);

	return SaveLog(users);
}
//-----------------------------------------------------------------------------------
bool CMyLogFile::SaveLog(char* names)
{
CString logpath;
CFile  LF;
char   errmsg[255];
CFileException ex;

	//get logging folder from the registry
	if(GetLogFolder(&logpath)==true)
		logpath = logpath + m_logfilename;
	else
		return false;

	if(LF.Open(logpath,CFile::modeWrite)!=1)
		if(LF.Open(logpath,CFile::modeWrite|CFile::modeCreate)!=1)
		{//Connot open or create file
```

```
                        ex.GetErrorMessage(errmsg,255);
                        ::MessageBox(m_pmaindlg->m_hWnd,errmsg,"Usage Meter",MB_OK|MB_ICONSTOP);
                        return false;
                }

        //Goto the end of the file
        LF.Seek((UINT)0,CFile::end);

char*  buff=NULL;
DWORD  l;
        l = IU_FILE_SRT_SIZE + m_us.names_len;
        buff = new char[l];
        memcpy(buff,&m_us,IU_FILE_SRT_SIZE);
        memcpy(buff+IU_FILE_SRT_SIZE,names,m_us.names_len);
        LF.Write(buff,l);
        LF.Close();
        delete[] buff;
        return true;
}
//-----------------------------------------------------------------------------------------
CString      CMyLogFile::CreateTitle()
{
CString title;
        title = NumToMothname(GetSystemMonth());
        //File name set's here
        m_logfilename = title;
        #ifdef _DEBUG
        title = "Usage Meter Debug (" + title + ")";
        #else
        title = "Usage Meter (" + title + ")";
        #endif
return title;
}
//-----------------------------------------------------------------------------------------
void CMyLogFile::MonthChange()
{
        m_pmaindlg->SetWindowText(CreateTitle());
        m_logfilename = m_logfilename + ".iuf";
}
//-----------------------------------------------------------------------------------------
void CMyLogFile::FormatTotal(char *buf, DWORD seconds)
{//Capacity <999:59:59
DWORD num,single,divid,index;
DWORD n[3];

        buf[3] = buf[6] = ':';

//Convert seconds to time
        n[0]          = seconds/3600;     //Hourse
        seconds             = seconds%3600; //Reminder
        n[1]          = seconds/60; //Minutes
        n[2]          = seconds%60; //Reminder(That's seconds)

for(int t=0;t<=2;t++)
{//Loop through three time components
        //Get each components
        num = n[t];

        if(t==0)
        {//Hour part can be > 59 so deferent algo
                for(int i=0;i<=2;i++)
                {
                        switch (i)
                        {
                        case 0:
                                divid=100;
                                single = num/divid;
                                break;
                        case 1:
                                divid=10;
                                num = single;
                                single = num/divid;
                                break;
```

```
                    }

                    switch (single)
                    {
                    case 0:
                           buf[i]='0';single = num;break;
                    case 1:
                           buf[i]='1';single=num-1*divid;break;
                    case 2:
                           buf[i]='2';single=num-2*divid;break;
                    case 3:
                           buf[i]='3';single=num-3*divid;break;
                    case 4:
                           buf[i]='4';single=num-4*divid;break;
                    case 5:
                           buf[i]='5';single=num-5*divid;break;
                    case 6:
                           buf[i]='6';single=num-6*divid;break;
                    case 7:
                           buf[i]='7';single=num-7*divid;break;
                    case 8:
                           buf[i]='8';single=num-8*divid;break;
                    case 9:
                           buf[i]='9';single=num-9*divid;break;
                    }
             }
      index = 4;
      }else{//Minuts and seconds always < 59

             for(int i=0;i<=1;i++)
             {
                    if(i==0)
                           single = num/10;

                    switch (single)
                    {
                    case 0:
                           buf[index]='0';single = num;break;
                    case 1:
                           buf[index]='1';single=num-1*10;break;
                    case 2:
                           buf[index]='2';single=num-2*10;break;
                    case 3:
                           buf[index]='3';single=num-3*10;break;
                    case 4:
                           buf[index]='4';single=num-4*10;break;
                    case 5:
                           buf[index]='5';single=num-5*10;break;
                    case 6:
                           buf[index]='6';single=num-6*10;break;
                    case 7:
                           buf[index]='7';single=num-7*10;break;
                    case 8:
                           buf[index]='8';single=num-8*10;break;
                    case 9:
                           buf[index]='9';single=num-9*10;break;
                    }
             index++;
             }
      index++;
      }
}
}
//--------------------------------------------------------------------------------

void CMyLogFile::SetStatusFile(WORD flag)
{
char   path[256];
DWORD  len=256;
HANDLE fileh;

      RegGet(HKEY_LOCAL_MACHINE,"LogFolder",path,&len);
```

```
	if((flag==CLEAR_STATUS)||(flag==CONNECTED)||(flag==DISCONNECTED))
	{	memcpy(path+(len-2),"Connected\0",10);
		DeleteFile(path);
		memcpy(path+(len-2),"Disconnected\0",13);
		DeleteFile(path);
	}

	if(flag==CONNECTED)
	{	memcpy(path+(len-2),"Connected\0",10);

	fileh=CreateFile(path,0,FILE_SHARE_READ,NULL,CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL);
		CloseHandle(fileh);
	}

	if(flag==DISCONNECTED)
	{	memcpy(path+(len-2),"Disconnected\0",13);

	fileh=CreateFile(path,0,FILE_SHARE_READ,NULL,CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL);
		CloseHandle(fileh);
	}
}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyLogFolder.h: interface for the CMyPath class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_MYPATH_H__B281C45E_BC3A_4A34_9950_9F2F77ADD7F5__INCLUDED_)
#define AFX_MYPATH_H__B281C45E_BC3A_4A34_9950_9F2F77ADD7F5__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "MyUser.h"

class CMyLogFolder : public CMyUser
{
public:
	DWORD* m_pprivilage;
		CMyLogFolder();
	virtual ~CMyLogFolder();
	bool   GetLogFolder(CString* plogfolder);
private:
	bool SetLogFolder(CString* plogfolder);
};

#endif // !defined(AFX_MYPATH_H__B281C45E_BC3A_4A34_9950_9F2F77ADD7F5__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyPath.cpp: implementation of the CMyLogFolder class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "usage meter.h"
#include "DlgPath.h"
#include "MyLogFolder.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
CMyLogFolder::CMyLogFolder(){}
CMyLogFolder::~CMyLogFolder(){}
//-----------------------------------------------------------------------------
bool CMyLogFolder::GetLogFolder(CString* plogfolder)
{
```

```
WIN32_FIND_DATA fd;
HANDLE              hfind;
char                path[256];
DWORD               len=256;

       RegGet(HKEY_LOCAL_MACHINE,"LogFolder",path,&len);

       //Check wether the folder is exist
       hfind = FindFirstFile(path, &fd);
       FindClose(hfind);
       if( (hfind==INVALID_HANDLE_VALUE) || (path[1]!=':') )
       {//Nop! folder is not there, so prompt for a new path
              if(SetLogFolder(plogfolder)==false)
                     return false;
              len = plogfolder->GetLength();
              memcpy(path,plogfolder->GetBuffer(len),len);
              len++;
       }
       //Remove '*'
       path[len-2] = '\0';
       *plogfolder = path;
       return true;
}
//-----------------------------------------------------------------------------------------
bool CMyLogFolder::SetLogFolder(CString* plogfolder)
{//Ask user to enter path
CDlgPath dlg_path;
       dlg_path.m_plogpath = plogfolder;
       dlg_path.m_pprivilage = m_pprivilage;
       dlg_path.DoModal();
       if(dlg_path.m_errer==true)
              return false;
       else
              return true;
}

|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyCounter.h: interface for the CMyCounter class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_MYCOUNTER_H__2FA33CAA_26B5_4CB5_BFDC_75D3F16E6894__INCLUDED_)
#define AFX_MYCOUNTER_H__2FA33CAA_26B5_4CB5_BFDC_75D3F16E6894__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "MyBeep.h"

class CMyCounter  : public CMyBeep
{
public:
       void   SetAt(char* time,int m_roughdu,bool resetbeep);
       void   Reset();

       void   Incriment(char* duration);
       //CString    GetRoughDuration();
       void   GetRoughDuration(char* duration);

       int          m_rough_sec;
       int          *m_pstart_sec;

private:
       int          m_refresh_index;
       WORD   m_i[3];
};

#endif // !defined(AFX_MYCOUNTER_H__2FA33CAA_26B5_4CB5_BFDC_75D3F16E6894__INCLUDED_)

|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
// MyCounter.cpp: implementation of the CMyCounter class.
//
//////////////////////////////////////////////////////////////////////
#include "Resource.h"
#include "stdafx.h"
#include "MyCounter.h"

//--------------------------------------------------------------------------------
void CMyCounter::Incriment(char *duration)
{
	m_i[S]++;
	m_rough_sec++;

	if(m_i[S]==60)
	{//Seconds
		m_i[S]=0;
		//Minutes
		m_i[M]++;
		m_refresh_index++;
			if(m_i[M]==60)
			{//Hours
				m_i[M]=0;
				m_i[H]++;
			}
	}

	//Refresh counter by recalculating duration using time diference
	//once after 5 min6
	if(m_refresh_index==5)
	{
		m_refresh_index=0;
		char now[9];char dur[9];;
		GetTime(now);
		TimeElapse(*m_pstart_sec,now,dur);
		SetAt(dur,m_rough_sec,false);
	}
	//Call Beep member of Beep class
	Beep();

	MakeTime(duration,m_i);
}
//--------------------------------------------------------------------------------
void CMyCounter::Reset()
{//Set counter to "00:00:00"
	ResetBeepCount();
	m_refresh_index = m_rough_sec = 0;
	m_i[H] = m_i[M] = m_i[S] = 0;
}
//--------------------------------------------------------------------------------
void CMyCounter::SetAt(char* time,int m_roughdu,bool resetbeep)
{//Set counter to specified time
int		itime[3];
int		values[2];
int		index;
char*	com;

	if(resetbeep==true)
		ResetBeepCount();
	m_rough_sec = m_roughdu;

	m_refresh_index = 0;
	index=0;

		for(int y=0;y<8;y=y+3)
		{//Loop though one time (23:45:34)
		com = time+y;
		//initialize
		itime[index]=0;
			for(int i=0;i<2;i++)
			{//Convert one time componemt (23)
				switch (com[i])
				{
				case '0':
```

```
                                        values[i]=0;break;
                                case '1':
                                        values[i]=1;break;
                                case '2':
                                        values[i]=2;break;
                                case '3':
                                        values[i]=3;break;
                                case '4':
                                        values[i]=4;break;
                                case '5':
                                        values[i]=5;break;
                                case '6':
                                        values[i]=6;break;
                                case '7':
                                        values[i]=7;break;
                                case '8':
                                        values[i]=8;break;
                                case '9':
                                        values[i]=9;break;
                                }
                                //Tenth place
                                if(i==0)
                                        values[0]=values[0]*10;

                                itime[index]=itime[index]+values[i];
                        }
                index++;
                }

        m_i[H]=itime[0];
        m_i[M]=itime[1];
        m_i[S]=itime[2];
}
//-------------------------------------------------------------------------------
/*CString     CMyCounter::GetRoughDuration()
{
CString temp;
char   du[9];
        memcpy(du,"00:00:00\0",9);
        GetRoughDuration(du);
        temp = du;
        return temp;
}*/
//-------------------------------------------------------------------------------
void CMyCounter::GetRoughDuration(char* duration)
{
        Format_Seconds(m_rough_sec,duration);
}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyListView.h: interface for the CMyListView class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_MYLISTVIEW_H__916B904B_6865_4D13_B1CD_7080D54812EE__INCLUDED_)
#define AFX_MYLISTVIEW_H__916B904B_6865_4D13_B1CD_7080D54812EE__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CMyListView
{
private:
        LVITEM m_li;
public:

        void SetupListview(HWND hwinList,HWND hintList,HFONT hwinfont,HFONT hintfont);
        void InsertRow(WORD index,char* lon_num,IU_FILE_STRUCT* us,char* user_name);
        void InsertTotal(WORD index,HWND hList,char* total);
        void DrawList(HWND hList,BOOL enable);
        void DeleteAll(HWND hList);
```

```
	virtual ~CMyListView();
	CMyListView();

	HWND   m_hlistview;
};

#endif // !defined(AFX_MYLISTVIEW_H__916B904B_6865_4D13_B1CD_7080D54812EE__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyListView.cpp: implementation of the CMyListView class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "usage meter.h"
#include "MyListView.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CMyListView::CMyListView()
{
	m_li.mask = LVIF_TEXT;
	m_li.state = NULL;
	m_li.stateMask  = NULL;
	m_li.iImage = NULL;
	m_li.lParam = NULL;
	m_li.iIndent = 0;
	m_li.cchTextMax = 0;
}

CMyListView::~CMyListView()
{

}

void CMyListView::InsertRow(WORD index,char* lon_num,IU_FILE_STRUCT* us,char* user_name)
{
	index--;
	//Log Number
	m_li.iSubItem= 0;
	m_li.iItem  = index;
	m_li.pszText  = lon_num;
	::SendMessage(m_hlistview,LVM_INSERTITEM,0,(LPARAM)&m_li);
	//Day
	m_li.iSubItem= 1;
	m_li.pszText  = us->day;
	::SendMessage(m_hlistview,LVM_SETITEMTEXT,(WPARAM)index,(LPARAM)&m_li);
	//From To
	m_li.iSubItem= 2;
	m_li.pszText  = us->start_end;
	::SendMessage(m_hlistview,LVM_SETITEMTEXT,(WPARAM)index,(LPARAM)&m_li);
	//Duration
	m_li.iSubItem= 3;
	m_li.pszText  = us->duration;
	::SendMessage(m_hlistview,LVM_SETITEMTEXT,(WPARAM)index,(LPARAM)&m_li);
	//User
	m_li.iSubItem= 4;
	m_li.cchTextMax     = us->names_len;
	m_li.pszText = user_name;
	::SendMessage(m_hlistview,LVM_SETITEMTEXT,(WPARAM)index,(LPARAM)&m_li);
	//Rough Duration
	m_li.iSubItem= 5;
	m_li.pszText  = us->roughdu;
	::SendMessage(m_hlistview,LVM_SETITEMTEXT,(WPARAM)index,(LPARAM)&m_li);
```

```
}
//-------------------------------------------------------------------------------
void CMyListView::DeleteAll(HWND hList)
{
	::SendMessage(hList,LVM_DELETEALLITEMS,0,0);
}
//-------------------------------------------------------------------------------
void CMyListView::DrawList(HWND hList,BOOL enable)
{
	::SendMessage(hList,WM_SETREDRAW ,(WPARAM)enable,0);
}
//-------------------------------------------------------------------------------
void CMyListView::InsertTotal(WORD index,HWND hList, char *total)
{
	//Log Number
	m_li.iSubItem= 0;
	m_li.iItem  = index;
	m_li.pszText = "";
	::SendMessage(hList,LVM_INSERTITEM,0,(LPARAM)&m_li);
	//Day
	m_li.iSubItem= 5;
	m_li.pszText = total;
	::SendMessage(hList,LVM_SETITEMTEXT,(WPARAM)index,(LPARAM)&m_li);
}
//-------------------------------------------------------------------------------
void CMyListView::SetupListview(HWND hwinList,HWND hintList,HFONT hwinfont,HFONT hintfont)
{
HWND		htemp;
LVCOLUMN	lco;

	lco.mask = LVCF_FMT|LVCF_TEXT|LVCF_WIDTH;
	lco.fmt = LVCFMT_CENTER;
    lco.cchTextMax = 0;
    lco.iImage = 0;
    lco.iOrder = 0;

	::SendMessage(hwinList,WM_SETFONT,(WPARAM)hwinfont,MAKELPARAM(FALSE, 0));
	::SendMessage(hintList,WM_SETFONT,(WPARAM)hintfont,MAKELPARAM(FALSE, 0));

for(int i=0;i<=1;i++)
{
	if(i==0)
		htemp = hintList;
	else
		htemp = hwinList;
	//Set Font
	::SendMessage(htemp,LVM_SETTEXTCOLOR,0,(LPARAM)RGB(0,0,255));
	//Set Extended styles
	::SendMessage(htemp,LVM_SETEXTENDEDLISTVIEWSTYLE,(WPARAM)LVS_EX_GRIDLINES|
LVS_EX_FULLROWSELECT,(LPARAM)LVS_EX_GRIDLINES|LVS_EX_FULLROWSELECT);
	//Prepair colunms
	lco.pszText = "Log";lco.cx = 40;
	::SendMessage(htemp,LVM_INSERTCOLUMN,(WPARAM)1,(LPARAM)&lco);
	lco.pszText = "Day";lco.cx = 40;
	::SendMessage(htemp,LVM_INSERTCOLUMN,(WPARAM)2,(LPARAM)&lco);
	lco.pszText = "From To";lco.cx = 145;
	::SendMessage(htemp,LVM_INSERTCOLUMN,(WPARAM)3,(LPARAM)&lco);
	lco.pszText = "Duration";lco.cx = 80;
	::SendMessage(htemp,LVM_INSERTCOLUMN,(WPARAM)4,(LPARAM)&lco);
	lco.pszText = "User(s)";lco.cx = 160;lco.fmt = LVCFMT_LEFT;
	::SendMessage(htemp,LVM_INSERTCOLUMN,(WPARAM)5,(LPARAM)&lco);
	lco.pszText = "Rough Duration";lco.cx = 105; ;lco.fmt = LVCFMT_CENTER;
	::SendMessage(htemp,LVM_INSERTCOLUMN,(WPARAM)6,(LPARAM)&lco);
	//Move ListView
	::MoveWindow(htemp,12,35,486,442,FALSE);
}


}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyMonthSyncro.h: interface for the CMyMonthSyncro class.
```

```
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_MYMONTHSYNCRO_H__2C321E6D_C6A7_4F11_8AC6_4C92367B985B__INCLUDED_)
#define AFX_MYMONTHSYNCRO_H__2C321E6D_C6A7_4F11_8AC6_4C92367B985B__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "MyTimeMath.h"

class CMyMonthSyncro : public CMyTimeMath
{
public:
	bool	DetectMonChange();
protected:
			CMyMonthSyncro();
	CString NumToMothname(int monnum);
	int		GetSystemMonth();
private:
	DWORD	m_regmonth;
};

#endif // !defined(AFX_MYMONTHSYNCRO_H__2C321E6D_C6A7_4F11_8AC6_4C92367B985B__INCLUDED_)

|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyMonthSyncro.cpp: implementation of the CMyMonthSyncro class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Usage Meter.h"
#include "MyMonthSyncro.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
CMyMonthSyncro::CMyMonthSyncro()
{
	m_regmonth=RegGet(HKEY_LOCAL_MACHINE,"Month");
}
//////////////////////////////////////////////////////////////////////
bool CMyMonthSyncro::DetectMonChange()
{
 SYSTEMTIME	ST;
 DWORD sys_mon;

	::GetLocalTime(&ST);
	sys_mon = ST.wMonth;
	if(sys_mon!=m_regmonth)
	{//month has changed
		//update registry
		m_regmonth = sys_mon;
		RegSet(HKEY_LOCAL_MACHINE,"Month",sys_mon);
		return true;
	}else{
		return false;
	}
}
//////////////////////////////////////////////////////////////////////
int CMyMonthSyncro::GetSystemMonth()
{
SYSTEMTIME ST;
	::GetLocalTime(&ST);
	return ST.wMonth;
}
//////////////////////////////////////////////////////////////////////
```

```
CString CMyMonthSyncro::NumToMothname(int monnum)
{
	switch(monnum)
	{
	case 1:
		return "January";
	case 2:
		return "February";
	case 3:
		return "March";
	case 4:
		return "April";
	case 5:
		return "May";
	case 6:
		return "June";
	case 7:
		return "July";
	case 8:
		return "August";
	case 9:
		return "September";
	case 10:
		return "October";
	case 11:
		return "November";
	case 12:
		return "December";
	}
	return "invalid";
}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyRegistry.h: interface for the CMyRegistry class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_MYREGISTRY_H__532C276B_C25D_402A_9CC3_885626E63C88__INCLUDED_)
#define AFX_MYREGISTRY_H__532C276B_C25D_402A_9CC3_885626E63C88__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "Resource.h"

class CMyRegistry
{
public:
	//char* SP(CString* ps);
		CMyRegistry();//constructor
void	RegSet(HKEY Hive,char* ValueName,DWORD Value);
void	RegSet(HKEY Hive,char* ValueName,char* Value,DWORD size);
DWORD	RegGet(HKEY Hive,char* ValueName);
void	RegGet(HKEY Hive,char* ValueName,char* chrval,DWORD* size);


private:
char*	m_reg_key;
HKEY	m_key_handle;
SECURITY_ATTRIBUTES m_secu_atri;
};
#endif // !defined(AFX_MYREGISTRY_H__532C276B_C25D_402A_9CC3_885626E63C88__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyRegistry.cpp: implementation of the CMyRegistry class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Usage Meter.h"
```

```
#include "MyRegistry.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////////////////////
CMyRegistry::CMyRegistry()
{
	#ifdef _DEBUG
	m_reg_key = "SOFTWARE\\Malwatta\\UsageMeter Debug";
	#else
	m_reg_key = "SOFTWARE\\Malwatta\\Usage Meter";
	#endif
	//Prepaire Security Attributes
	m_secu_atri.nLength=sizeof(m_secu_atri);
	m_secu_atri.bInheritHandle=1;
	m_secu_atri.lpSecurityDescriptor=NULL;
}
//////////////////////////////////////////////////////////////////////////////////////
void CMyRegistry::RegSet(HKEY Hive,char* ValueName,DWORD Value)
{//DWORD
	//Open or Create Key
	RegCreateKeyEx(Hive,m_reg_key,NULL,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,
&m_secu_atri, &m_key_handle,NULL);
	//Set Value
	RegSetValueEx(m_key_handle,ValueName,0,REG_DWORD,(LPBYTE)&Value, sizeof(Value));
	//Close
	RegCloseKey(m_key_handle);
}
//////////////////////////////////////////////////////////////////////////////////////
void CMyRegistry::RegSet(HKEY Hive,char* ValueName,char* Value,DWORD size)
{//STRING
	//Open or Create Key
	RegCreateKeyEx(Hive,m_reg_key,NULL,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,
&m_secu_atri, &m_key_handle,NULL);
	//Set Value
	RegSetValueEx(m_key_handle,ValueName,0,REG_SZ,(LPBYTE)Value, size);
	//Close
	RegCloseKey(m_key_handle);
}
//////////////////////////////////////////////////////////////////////////////////////
DWORD CMyRegistry::RegGet(HKEY Hive,char* ValueName)
{//DWORD
DWORD size = 4;
BYTE Value[4];
	//Open or Create Key
	RegCreateKeyEx(Hive, m_reg_key,NULL,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,
&m_secu_atri, &m_key_handle,NULL);
	//Get Value
	if(RegQueryValueEx(m_key_handle,ValueName ,NULL,NULL,Value,&size)!=0)
		return 0;
	//Close
	RegCloseKey(m_key_handle);
	return *(DWORD*)Value;
}
//////////////////////////////////////////////////////////////////////////////////////
void CMyRegistry::RegGet(HKEY Hive,char* ValueName,char* chrval,DWORD* size)
{//CHAR

	//Open or Create Key
	RegCreateKeyEx(Hive, m_reg_key,NULL,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,
&m_secu_atri, &m_key_handle,NULL);
	//Get Value
	if(RegQueryValueEx(m_key_handle,ValueName ,NULL,NULL,(BYTE*)chrval,size)!=0)
	{
		RegCloseKey(m_key_handle);
		return;
	}
	//Close
	RegCloseKey(m_key_handle);
```

```
}

/*char* CMyRegistry::SP(CString *ps)
{
	return ps->GetBuffer(ps->GetLength());
}*/

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyTimeMath.h: interface for the CMyTimeMath class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_MYTIMEMATH_H__A851DB42_2986_461F_946A_9A1837FBCD19__INCLUDED_)
#define AFX_MYTIMEMATH_H__A851DB42_2986_461F_946A_9A1837FBCD19__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include  "MyRegistry.h"

struct DAY_STRUCT
{
char   day[3];
WORD   daylen;
};

class CMyTimeMath : public CMyRegistry
{
public:
	void   TimeElapse(DWORD start_sec,char* end_time,char* elapse);
	void   GetDay(DAY_STRUCT* today,bool* daychanged=NULL,char* preday=NULL);
	void   TimeAddition(int* total_seconds,char* add_time);
	//void Format_Seconds(int seconds,CString* time);
	void   Format_Seconds(int seconds, char *time);
	void   NumToString(char* c,int n);
	void   MakeTime(char *buf, WORD* n);
	//int        ToSeconds(CString* time);
	int          ToSeconds(char *time);
	void   GetTime(char* time);
	//CString GetTime();
		CMyTimeMath();

private:
	bool CompareBuffer(char* one,char* two,WORD size);
	CString sH,sM,sS;
	char m_previ_day[3];
};

#endif // !defined(AFX_MYTIMEMATH_H__A851DB42_2986_461F_946A_9A1837FBCD19__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyTimeMath.cpp: implementation of the CMyTimeMath class.

#include "stdafx.h"
#include "Usage Meter.h"
#include "MyTimeMath.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

CMyTimeMath::CMyTimeMath()
{
DAY_STRUCT temp;
	GetDay(&temp);
	memcpy(m_previ_day,temp.day,2);
}
//--------------------------------------------------------------------------------
```

```
void CMyTimeMath::TimeAddition(int* total_seconds,char* add_time)
{//time recive like "12:45:34"
//More efficient function to convert time component to Int(eg - 23:45:56)
int itime[3];
int values[2];
int index;
char* com;

	index=0;

		for(WORD y=0;y<8;y=y+3)
		{//Loop though one time (23:45:34)
		com = add_time+y;
		//initialize
		itime[index]=0;
			for(WORD i=0;i<2;i++)
			{//Convert one time componemt (23)
				switch (com[i])
				{
				case '0':
					values[i]=0;break;
				case '1':
					values[i]=1;break;
				case '2':
					values[i]=2;break;
				case '3':
					values[i]=3;break;
				case '4':
					values[i]=4;break;
				case '5':
					values[i]=5;break;
				case '6':
					values[i]=6;break;
				case '7':
					values[i]=7;break;
				case '8':
					values[i]=8;break;
				case '9':
					values[i]=9;break;
				}
				//Tenth place
				if(i==0)
					values[0]=values[0]*10;
				itime[index]=itime[index]+values[i];
			}
		index++;
		}

	//End time convert to seconds
	itime[M] = itime[M] * 60;
	itime[M] = itime[M] + itime[S];
	itime[H] = itime[H] * 3600;
	itime[H] = itime[H] + itime[M];

	//add seconds
	*total_seconds = *total_seconds + itime[H];
}
//--------------------------------------------------------------------------------
void CMyTimeMath::TimeElapse(DWORD start_sec,char* end_time,char* elapse)
{
DWORD end_sec;
DWORD elapse_sec;
WORD i[3];

	elapse_sec = 0;
	end_sec = ToSeconds(end_time);

	//calculate Time Elaps
	if(end_sec>start_sec){
	//not pass Mid Night
	elapse_sec = end_sec - start_sec;
	}else{
	//pass Mid Night
```

```
      elapse_sec = 86400 - start_sec;//Until mid night
    elapse_sec = elapse_sec + end_sec;//Rest (After mid night)
      }

//Convert seconds to time (Time Elaps)
#pragma warning(disable:4244)
      i[H]         = elapse_sec/3600;  //Hourse
      elapse_sec   = elapse_sec%3600;  //Reminder
      i[M]         = elapse_sec/60;    //Minutes
      i[S    ]           = elapse_sec%60;    //Reminder(That's seconds)
#pragma warning(default:4244)

//Format output
      MakeTime(elapse,i);
}
//-----------------------------------------------------------------------------------
void CMyTimeMath::GetDay(DAY_STRUCT* today,bool* daychanged,char* preday)
{//Get Day
SYSTEMTIME   st;

      ZeroMemory(today->day,2);
      GetLocalTime(&st);
      NumToString(today->day, st.wDay);

      if(daychanged!=NULL)
      {//valid pointer
            if(CompareBuffer(today->day,m_previ_day,2)==false)
            {//day changed
                  //return previous day
                  memcpy(preday,m_previ_day,2);
                  memcpy(m_previ_day,today->day,2);
                  *daychanged=true;
            }else{
                  *daychanged=false;
            }
      }
      //Return the lenth of day
      if(st.wDay<9)
      {
            //today->day [1] = '\0';
            today->daylen = 1;
      }else{
            //today->day[2] = '\0';
            today->daylen = 2;
      }
}
//-----------------------------------------------------------------------------------
void CMyTimeMath::GetTime(char *time)
{
SYSTEMTIME   st;
WORD         t[3];

      time[2] = time[5] = ':';
      time[8] = '\0';

      GetLocalTime(&st);

      t[0] = st.wHour;
      t[1] = st.wMinute;
      t[2] = st.wSecond;

      MakeTime(time,t);
}
//-----------------------------------------------------------------------------------
/*void CMyTimeMath::Format_Seconds(int seconds, CString *time)
{//Capacity <99:59:59
      *time = "00:00:00";
      Format_Seconds(seconds,SP(time));
}*/
//-----------------------------------------------------------------------------------
void CMyTimeMath::Format_Seconds(int seconds, char *time)
{//Capacity <99:59:59
WORD i[3];
```

```
//Convert seconds to time
	i[H]		= seconds/3600;		//Hourse
	seconds			= seconds%3600; //Reminder
	i[M]		= seconds/60;//Minutes
	i[S]		= seconds%60;//Reminder(That's seconds)
//Convert seconds to time

//Format output
	MakeTime(time,i);
}
//-------------------------------------------------------------------------------------
/*int CMyTimeMath::ToSeconds(CString *time)
{
	return ToSeconds(SP(time));
}*/
//-------------------------------------------------------------------------------------
int CMyTimeMath::ToSeconds(char *time)
{
int itime[3];
int values[2];
int index;
char* com;

	index=0;

		for(WORD y=0;y<8;y=y+3)
		{//Loop though one time (23:45:34)
		com = time+y;
		//initialize
		itime[index]=0;
			for(WORD i=0;i<2;i++)
			{//Convert one time componemt (23)
				switch (com[i])
				{
				case '0':
					values[i]=0;break;
				case '1':
					values[i]=1;break;
				case '2':
					values[i]=2;break;
				case '3':
					values[i]=3;break;
				case '4':
					values[i]=4;break;
				case '5':
					values[i]=5;break;
				case '6':
					values[i]=6;break;
				case '7':
					values[i]=7;break;
				case '8':
					values[i]=8;break;
				case '9':
					values[i]=9;break;
				}
				//Tenth place
				if(i==0)
					values[0]=values[0]*10;

				itime[index]=itime[index]+values[i];
			}
		index++;
		}

	//End time convert to seconds
	itime[M] = itime[M] * 60;
	itime[M] = itime[M] + itime[S];
	itime[H] = itime[H] * 3600;
	itime[H] = itime[H] + itime[M];
return itime[H];
}

//-------------------------------------------------------------------------------------
```

```
void CMyTimeMath::MakeTime(char *buf, WORD* n)
{//Capacity <99:59:59
WORD num,single,index;

	index = 0;

	for(WORD t=0;t<=2;t++)
	{//Loop through three time components
	//Get each components
	num = n[t];
		for(WORD i=0;i<=1;i++)
		{//Convert each component to char
			if(i==0)
				single = num/10;

			switch (single)
			{
			case 0:
				buf[index]='0';single = num;break;
			case 1:
				buf[index]='1';single=num-1*10;break;
			case 2:
				buf[index]='2';single=num-2*10;break;
			case 3:
				buf[index]='3';single=num-3*10;break;
			case 4:
				buf[index]='4';single=num-4*10;break;
			case 5:
				buf[index]='5';single=num-5*10;break;
			case 6:
				buf[index]='6';single=num-6*10;break;
			case 7:
				buf[index]='7';single=num-7*10;break;
			case 8:
				buf[index]='8';single=num-8*10;break;
			case 9:
				buf[index]='9';single=num-9*10;break;
			}
		index++;
		}
	index++;
	}
}
//--------------------------------------------------------------------------------
void CMyTimeMath::NumToString(char *buf, int n)
{//Capacity <999
int	num,min,sav;
	sav = n;

for(int i=0;i<=2;i++)
{
	switch (i)
	{
	case 0:
		min=100;
		num = n/min;
		break;
	case 1:
		min=10;
		n = num;
		num = n/min;
		break;
	}

	switch (num)
	{
	case 0:
		buf[i]='0';num = n;break;
	case 1:
		buf[i]='1';num=n-1*min;break;
	case 2:
		buf[i]='2';num=n-2*min;break;
	case 3:
```

```
			buf[i]='3';num=n-3*min;break;
		case 4:
			buf[i]='4';num=n-4*min;break;
		case 5:
			buf[i]='5';num=n-5*min;break;
		case 6:
			buf[i]='6';num=n-6*min;break;
		case 7:
			buf[i]='7';num=n-7*min;break;
		case 8:
			buf[i]='8';num=n-8*min;break;
		case 9:
			buf[i]='9';num=n-9*min;break;
		}
}

		if(sav<100)
		{
			if(sav<10)
			{//Like 34
				buf[0] = buf[2];
				buf[1] = '\0';
			}else{//Like 6
				buf[0] = buf[1];
				buf[1] = buf[2];
				buf[2] = '\0';
			}
		}else//Like 624
			buf[3] = '\0';
}
//--------------------------------------------------------------------------------------
bool CMyTimeMath::CompareBuffer(char *one, char *two, WORD size)
{
		size--;
		for(WORD i=0;i<=size;i++)
		{
			if(one[i]!=two[i])
				return false;
		}
return true;
}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyUser.h: interface for the CMyUser class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_MYUSER_H__0EE5ADCE_230A_4D20_AEE4_A955CC7D07BB__INCLUDED_)
#define AFX_MYUSER_H__0EE5ADCE_230A_4D20_AEE4_A955CC7D07BB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "MyMonthSyncro.h"

class CMyUser : public CMyMonthSyncro
{
public:
		void	InitializeUserList();
		void	RemoveUser();
		void	AddUser();
protected:
		CString*	Split(char *data, char delimit,DWORD* datalen,WORD arrsize);
		void		GetIntUsers(char* users,DWORD* size);
		void		GetWinUsers(char* users,DWORD* len);
private:
		CString* ExpandStrArr(CString *str_arr,WORD* size);
		void	GetCurrentUser(CString* user);
};

#endif // !defined(AFX_MYUSER_H__0EE5ADCE_230A_4D20_AEE4_A955CC7D07BB__INCLUDED_)
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyUser.cpp: implementation of the CMyUser class.

#include "stdafx.h"
#include "MyUser.h"

//-----------------------------------------------------------------------------
void CMyUser::GetCurrentUser(CString* user)
{//get current user name form windows
char   c[100];
DWORD  i=100;
       if(GetUserName(c,&i)!=0)
              *user = c;
       else
              *user = "Unknown";
}
//-----------------------------------------------------------------------------
void CMyUser::GetIntUsers(char* users,DWORD* size)
{//get reg saved user names
       RegGet(HKEY_LOCAL_MACHINE,"ActiveUsers",users,size);
}
//-----------------------------------------------------------------------------
void CMyUser::GetWinUsers(char* users,DWORD* len)
{//get reg saved user names
       RegGet(HKEY_LOCAL_MACHINE,"LogonUsers",users,len);
}
//-----------------------------------------------------------------------------
void CMyUser::AddUser()
{//add users to registry
CString             curuser,users,regusers;
CString*     userarr;
DWORD        size;
bool         ex_user=false;

//LOGON USERS**************************
       //get user list from registry
char   czusers[100];
       size = 100;
       RegGet(HKEY_LOCAL_MACHINE,"LogonUsers",czusers,&size);
       regusers = czusers;
       GetCurrentUser(&curuser);
       //check wether the user name already exist
       size = regusers.GetLength();
       userarr = Split(regusers.GetBuffer(size),',',&size,10);
       for(DWORD i=0;i<=size;i++)
       {
              if(userarr[i]==curuser)
              {
              ex_user = true;
              break;
              }
       }
       delete[] userarr;
       if(ex_user==false)
       {//nop add this user
              regusers = regusers  + ',' + curuser;
              size = regusers.GetLength();
              RegSet(HKEY_LOCAL_MACHINE,"LogonUsers",regusers.GetBuffer(size),size);
       }
//LOGON USERS**************************

//ACTIVE USERS*************************
       RegGet(HKEY_LOCAL_MACHINE,"ActiveUsers",czusers,&size);
       regusers = czusers;
       regusers = regusers  + ',' + curuser;
       size = regusers.GetLength();
       RegSet(HKEY_LOCAL_MACHINE,"ActiveUsers",regusers.GetBuffer(size),size);
//ACTIVE USERS*************************
}
//-----------------------------------------------------------------------------
void CMyUser::RemoveUser()
```

```
{
CString             regusers ,curuser;
CString*     userarr;
DWORD        size;
char         users[100];

       RegGet(HKEY_LOCAL_MACHINE,"ActiveUsers",users,&size);
       GetCurrentUser(&curuser);

       userarr = Split(users,',',&size,10);

       for(DWORD i=0;i<=size;i++)
       {
              if(userarr[i]!=curuser)
              {
                     if(regusers.IsEmpty())
                     regusers = userarr[i];
                     else
                     regusers = regusers + ',' + userarr[i];
              }
       }
       delete[] userarr;
size = regusers.GetLength();
       RegSet(HKEY_LOCAL_MACHINE,"ActiveUsers",regusers.GetBuffer(size),size);
}
//--------------------------------------------------------------------------------
void CMyUser::InitializeUserList()
{
CString name;
DWORD  size;
       GetCurrentUser(&name);
       size = name.GetLength();
       RegSet(HKEY_LOCAL_MACHINE,"LogonUsers",name.GetBuffer(size),size);
       RegSet(HKEY_LOCAL_MACHINE,"ActiveUsers",name.GetBuffer(size),size);
}
//--------------------------------------------------------------------------------
CString* CMyUser::Split(char *data, char delimit,DWORD* datalen,WORD arrsize)
{
DWORD        index,p,l,oi;
char         temp_arr[200];
CString*     strarr;

       //create dynamic string array
       strarr = new CString[arrsize];

       oi = index = 0;
       *datalen = *datalen-1;//Reduce one as zerobase index

       for(DWORD i=0;i<=*datalen;i++)
       {
              if(data[i]==delimit)
              {
                     p = oi;
                     l = i - oi;
                     memcpy(temp_arr,data+p, l);
                     temp_arr[l] = '\0';
                     strarr[index++]=temp_arr;
                     oi = i+1;
                     if(index==arrsize)//Array should resize
                            strarr = ExpandStrArr(strarr,&arrsize);
              }
       }
       //Final item
       memcpy(temp_arr,data+oi, i-oi);
       temp_arr[i-oi] = '\0';
       strarr[index]=temp_arr;

       //Return the number of eliments in the array by pointer
       if(index!=0)
              *datalen = index - 1;
       else
              *datalen = 0;
       //return array
```

```
	return strarr;
}
//--------------------------------------------------------------------------------
CString* CMyUser::ExpandStrArr(CString *str_arr,WORD* size)
{
CString* new_strarr;

	new_strarr = new CString[*size+100];
	*size = *size-1;

	for(int i=0;i<=*size;i++)
		new_strarr[i] = str_arr[i];

	//delete old array
	delete[] str_arr;
	*size = *size+101;
	return new_strarr;
}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyClientPort.h: interface for the CMyClientPort class.
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_MYCLIENTPORT_H__44917915_42A4_41B4_BC44_6DCCD69AA0CB__INCLUDED_)
#define AFX_MYCLIENTPORT_H__44917915_42A4_41B4_BC44_6DCCD69AA0CB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "MyLogFile.h"
#include "MyCounter.h"

class CMyClientPort : public CMyLogFile
{
public:
	//fun
	void	IsServerRunning(char* winstart,DWORD* winroughdu);
	void	GrantPrivilages(HWND main_dlg);
protected:
	//ver
	DWORD	m_client_count;
	DWORD*	m_pprivilage;//pointer to MainDlg m_privilage
	DWORD	m_id;
};

#endif // !defined(AFX_MYCLIENTPORT_H__44917915_42A4_41B4_BC44_6DCCD69AA0CB__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyClientPort.cpp: implementation of the CMyClientPort class.
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Usage Meter.h"
#include "MyClientPort.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//--------------------------------------------------------------------------------
void CMyClientPort::IsServerRunning(char* winstart,DWORD* winroughdu)
{
	//ask for run mode
	RegSet(HKEY_LOCAL_MACHINE,"ServerPort",MSG_IS_SERVER_RUNNING);
	::Sleep(1100);

	//check whether a answer given
DWORD ans = RegGet(HKEY_LOCAL_MACHINE,"ServerPort");
```

```
	if(ans==MSG_YES_RUNNUNG)
	{//server given a answer (I am a client)
		*m_pprivilage = CLIENT;
		//get my ID
		m_id=RegGet(HKEY_LOCAL_MACHINE,"ID");
		//get start times
		DWORD size = 9;
		RegGet(HKEY_LOCAL_MACHINE,"WinStartTime",winstart,&size);
		//get rough duration
		*winroughdu = RegGet(HKEY_LOCAL_MACHINE,"WinRoughDura");
	}else{//no answer (I am a server)
		*m_pprivilage = SERVER;
	}

	//invalidate ServerPort
	RegSet(HKEY_LOCAL_MACHINE,"ServerPort",MSG_INVALID);
}
//--------------------------------------------------------------------------------
void CMyClientPort::GrantPrivilages(HWND main_dlg)
{
	//ask for privilages
	DWORD answer = RegGet(HKEY_LOCAL_MACHINE,"ServerPort");

	if(answer==MSG_GET_PREVILIGE)
	{//server given privilages
		DWORD id = RegGet(HKEY_LOCAL_MACHINE,"ID");
		if(m_id==id)
		{//given to me
			//Invalidate ServerPort
			RegSet(HKEY_LOCAL_MACHINE,"ServerPort",MSG_INVALID);
			//kill server comunication
			KillTimer(main_dlg,TIMER_SERVER_LINK);
			//get client count
			m_client_count=RegGet(HKEY_LOCAL_MACHINE,"ClientCount");
			//say to server that I got privillages
			RegSet(HKEY_LOCAL_MACHINE,"ClientPort",MSG_GOT_PREVILIGE);
			//assign privilage
			*m_pprivilage = SERVER;
		}
	}
}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyServerPort.h: interface for the CMyLinker class.
//

#if !defined(AFX_MYEMULATOR_H__298A481D_BBE8_4839_8631_5AD1A503F796__INCLUDED_)
#define AFX_MYEMULATOR_H__298A481D_BBE8_4839_8631_5AD1A503F796__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "MyClientPort.h"

class CMyServerPort : public CMyClientPort
{
public:
	void	IntServerPortObj(DWORD* prilageref);
	bool	GivePrivilage();
	void	SayServerActive(char* winstart,DWORD winroughdu);
};

#endif // !defined(AFX_MYEMULATOR_H__298A481D_BBE8_4839_8631_5AD1A503F796__INCLUDED_)

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

// MyServerPort.cpp: implementation of the CMyServerPort class.
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Usage Meter.h"
```

```
#include "MyServerPort.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//-------------------------------------------------------------------------------
void CMyServerPort::SayServerActive(char* winstart,DWORD winroughdu)
{
	DWORD qustion = RegGet(HKEY_LOCAL_MACHINE,"ServerPort");

	if(qustion==MSG_IS_SERVER_RUNNING)
	{//ACTUALY ASKED A QUSTION
		//give ID
		m_client_count++;
		RegSet(HKEY_LOCAL_MACHINE,"ID",m_client_count);
		//give win start times
		RegSet(HKEY_LOCAL_MACHINE,"WinStartTime",winstart,8);
		//give win rough duration times
		RegSet(HKEY_LOCAL_MACHINE,"WinRoughDura",winroughdu);
		//supply answer
		RegSet(HKEY_LOCAL_MACHINE,"ServerPort",MSG_YES_RUNNUNG);
	}
}
//-------------------------------------------------------------------------------
void CMyServerPort::IntServerPortObj(DWORD* prilageref)
{
	m_pprivilage=prilageref;
	m_client_count=0;
	m_id=0;
}
//-------------------------------------------------------------------------------
bool CMyServerPort::GivePrivilage()
{
	//reset client port
	RegSet(HKEY_LOCAL_MACHINE,"ClientPort",MSG_INVALID);

	for(DWORD i=1;i<=m_client_count;i++)
	{
		//give client cout
		RegSet(HKEY_LOCAL_MACHINE,"ClientCount",m_client_count);
		//who can get the privilages
		RegSet(HKEY_LOCAL_MACHINE,"ID",i);
		//give privilage to client
		RegSet(HKEY_LOCAL_MACHINE,"ServerPort",MSG_GET_PREVILIGE);
		::Sleep(600);

		if(MSG_GOT_PREVILIGE==RegGet(HKEY_LOCAL_MACHINE,"ClientPort"))
		{//some one got the privilage
			return true;
		}
	}

	return false;
}
```