```
m_ListCtrl->ModifyStyleEx(0, WS_EX_CLIENTEDGE);
AddExStyle(LVS_EX_FULLROWSELECT | LVS_OWNERDRAWFIXED);

int i;
LV_COLUMN lvc;

lvc.mask = LVCF_FMT | LVCF_WIDTH | LVCF_TEXT | LVCF_SUBITEM;
CString strTemp[2] = {"", ""};
int size[2] = {140,40};
for(i = 0; i < 2; i++)
{
    lvc.iSubItem = i;
    lvc.pszText = (char*)(LPCTSTR)strTemp[i];
    lvc.cx = size[i];
    lvc.fmt = LVCFMT_LEFT;
    m_ListCtrl->InsertColumn(i, &lvc);
}

pnid.cbSize = sizeof(NOTIFYICONDATA);
pnid.hIcon = LoadIcon(AfxGetApp()->
m_hInstance,MAKEINTRESOURCE(IDR_MAINFRAME));
pnid.hWnd = this->m_hWnd ;
sprintf(pnid.szTip, "聊天程序\n");
pnid.uCallbackMessage = WM_SYSTRAY;
pnid.uFlags = NIF_ICON | NIF_MESSAGE | NIF_TIP;
pnid.uID = ID_SYSTRAY;
Shell_NotifyIcon(NIM_ADD, &pnid);


if(ConnectToServer())
{
    m_bOnline = TRUE;
    m_pMsg->From = Name;
```

```
            m_pMsg->To = "All";
            m_pMsg->Type = 5;
            m_pMsg->ShortMessage = "Hello";
            m_pSocket->SendMsg(m_pMsg);
        }
        else
            m_bOnline = FALSE;

        CString m_strTemp;
        this->GetWindowText(m_strTemp);
        m_strTemp += "-";
        m_strTemp += Name;
        this->SetWindowText(m_strTemp);
        return 0;
    }

    void CUserDlg::AddExStyle(DWORD dwNewStyle)
    {
        DWORD dwStyle = ::SendMessage (m_ListCtrl->m_hWnd,
        LVM_GETEXTENDEDLISTVIEWSTYLE, 0, 0);
        dwStyle |= dwNewStyle;

        ::SendMessage (m_ListCtrl->m_hWnd,
        LVM_SETEXTENDEDLISTVIEWSTYLE, 0, dwStyle);

    }

    void CUserDlg::OnClose()
    {
        // TODO: Add your message handler code here and/or call default

        if(!m_pSocket)
        {
            delete m_pSocket;
```

```
        m_pSocket = NULL;
    }
    if (m_pMsg != NULL)
        delete m_pMsg;
    CDialog::OnClose();
}

BOOL CUserDlg::ConnectToServer()
{
    if (m_bOnline)
        return TRUE;

    m_pSocket = new CServiceSocket(this);

    if (m_pSocket == NULL)
    {
        AfxMessageBox("Couldn't allocate memory for service
        socket!");
        return FALSE;
    }

    if (!m_pSocket->Create())
    {
        delete m_pSocket;
        m_pSocket = NULL;
        AfxMessageBox("Create Socket Error!");
        return FALSE;
    }

    m_pSocket->Name = Name;
    //m_strAddress = "127.0.0.1";
    //m_nPort = 6666;

    while (!m_pSocket->Connect(m_strAddress, m_nPort))
```

```
    {
        if (AfxMessageBox("Retry again?",MB_YESNO) == IDNO)
        {
            delete m_pSocket;
            m_pSocket = NULL;
            return FALSE;
        }
    }

    m_pSocket->Init();

    m_pMsg->From = Name;
    m_pMsg->To = "All";
    m_pMsg->Type = 1;
    m_pMsg->ShortMessage = "Come in....";

    m_pSocket->SendMsg(m_pMsg);

    m_bOnline = TRUE;

    return TRUE;

}

void CUserDlg::ProcessReadMessage()
{
    do
    {
        ReceiveMsg();
        if (m_pSocket == NULL)
            return;
    }
    while(!m_pSocket->m_pArchiveIn->IsBufferEmpty());
```

```
}

void CUserDlg::ReceiveMsg()
{
    CString m_strTemp;
    int m_nPosition ;
    int m_nPrePos ;

    m_pMsg->Serialize(*m_pSocket->m_pArchiveIn);
    switch(m_pMsg->Type)
    {
    case 1:
        m_ListCtrl->AddItem(1, m_pMsg->From.GetBuffer(0), "");
        break;
    case 2:
        m_ListCtrl->Remove(m_pMsg->From.GetBuffer(0) );
        break;
    case 3:
        DisplayMsg();
        break;
    case 4:
        DisplayMsg();
        break;
    case 5:
        m_ListCtrl->DeleteAllItems();
        m_nPosition = -1;
        m_nPrePos = 0;
        m_nPosition = m_pMsg->ShortMessage.Find("##",0);
        while(m_nPosition != -1)
        {
            m_strTemp = m_pMsg->ShortMessage.Mid(m_nPrePos,
            m_nPosition - m_nPrePos);
            if(m_strTemp.Compare(Name))
            {
```

```
                m_ListCtrl->AddItem(1, m_strTemp.GetBuffer(0), "");
            }
            m_nPrePos = m_nPosition + 2;
            m_nPosition = m_pMsg->ShortMessage.Find("##",
            m_nPrePos);

        }

        break;
    case 6:
        break;
    case 7:
        break;
    default:
        break;
    }

}

void CUserDlg::SendMsg()
{
    CMsgDlg m_dlgMsg;
    m_dlgMsg.m_bSend = TRUE;
    m_dlgMsg.m_strFrom = Name;
    m_dlgMsg.m_strTo = m_pMsg->To;
    if(m_dlgMsg.DoModal() == IDOK)
    {
        m_pMsg->ShortMessage = m_dlgMsg.m_strMsg ;
        m_pMsg->Type = 3;
        m_pSocket->SendMsg(m_pMsg);
    }
}

void CUserDlg::DisplayMsg()
```

```
{
    CMsgDlg m_dlgMsg;
    CString m_strTemp;

    if(m_pMsg->From == "")
        return;
    m_strTemp = m_pMsg->From;
    m_dlgMsg.m_bSend = FALSE;
    m_dlgMsg.m_strTo = m_pMsg->To;
    m_dlgMsg.m_strMsg = m_pMsg->ShortMessage ;
    m_dlgMsg.m_strFrom = m_pMsg->From ;

    if(m_dlgMsg.DoModal() == IDOK)
    {
        m_pMsg->From = Name;
        //m_pMsg->To = m_dlgMsg.m_strTo ;
        m_pMsg->To = m_strTemp;
        m_pMsg->ShortMessage = m_dlgMsg.m_strMsg ;
        m_pMsg->Type = 3;
        m_pSocket->SendMsg(m_pMsg);
    }

}

void CUserDlg::OnAddMember(WPARAM wParam, LPARAM lParam)
{
    CString* pStr = (CString*)wParam;
    m_ListCtrl->AddItem((short)lParam, pStr->GetBuffer(0), NULL);
    pStr->ReleaseBuffer();
}

void CUserDlg::OnRemoveMember(WPARAM wParam, LPARAM lParam)
{
    CString* pStr = (CString*)wParam;
```

```
    m_ListCtrl->Remove(pStr->GetBuffer(0));
    pStr->ReleaseBuffer();
}

void CUserDlg::OnDblClickItem(WPARAM wParam, LPARAM lParam)
{
    CString* pStr = (CString*)lParam;
    m_pMsg->To = *pStr;
    SendMsg();
    //pStr->ReleaseBuffer();

}

void CUserDlg::OnCloseDlg()
{
    // TODO: Add your command handler code here
    SendCloseMsg();
    OnClose();
}

void CUserDlg::OnOffline()
{
    // TODO: Add your command handler code here
    if (!m_bOnline)
        return;
    m_pSocket->Close();
    if (!m_pSocket)
        delete m_pSocket;
    m_bOnline = FALSE;
}

void CUserDlg::OnOnline()
{
    // TODO: Add your command handler code here
```

```
    if (!m_bOnline)
    {
        ConnectToServer();
        m_bOnline = TRUE;
    }

}

void CUserDlg::OnRestore()
{
    // TODO: Add your command handler code here
    ShowWindow(SW_SHOW);

}

void CUserDlg::OnSendmsg()
{
    // TODO: Add your command handler code here
    m_ListCtrl->SendNotifyMessage(NM_DBLCLK,0,0);
}

void CUserDlg::OnSysTray(WPARAM wParam,LPARAM lParam)
{
    CPoint pt;
    CClientApp* app = (CClientApp*)AfxGetApp();
    switch (lParam) {
    case WM_RBUTTONUP:
    { // Let's track a popup menu
        GetCursorPos(&pt);
        HMENU hmenu = LoadMenu(AfxGetApp()->
        m_hInstance,MAKEINTRESOURCE(IDR_MENUTRAY));
        HMENU hpopup = GetSubMenu(hmenu, 0);
        switch (TrackPopupMenu(hpopup,// Popup menu to track
            TPM_RETURNCMD ,    // Return menu code
```

```
            TPM_RIGHTBUTTON,   // Track right mouse button?
            pt.x, pt.y,        // screen coordinates
            0,                 // reserved
            this->m_hWnd,            // owner
            NULL))             // LPRECT user can click in
            // without dismissing menu
        {
        case ID_CLOSE: //DestroyWindow(hwnd); break;
            SendCloseMsg();
            PostQuitMessage(0);
            Shell_NotifyIcon(NIM_DELETE, &pnid);
            break;
        case ID_RESTORE:
            OnRestore();
            break;
        case ID_ONLINE:
            OnOnline();
            break;
        case ID_OFFLINE:
            OnOffline();
            break;
        }
    }
    case WM_LBUTTONDBLCLK:
        ShowWindow(SW_SHOW);
    }
}

void CUserDlg::SendCloseMsg()
{
    if(!m_pSocket)
    {
        m_pMsg->From = Name;
        m_pMsg->To = "All";
```