```
#define UNICODE
#include<windows.h>
#include"AutomationServerWithRegFile.h"
// global function declarations
LRESULT CALLBACK WndProc(HWND,UINT,WPARAM,LPARAM);
// WinMain
int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine,int nCmdShow)
{
    // variable declarations
    WNDCLASSEX wndclass;
    HWND hwnd;
    MSG msg;
    TCHAR AppName[]=TEXT("Client");
    // code
    wndclass.cbSize=sizeof(wndclass);
    wndclass.style=CS_HREDRAW|CS_VREDRAW;
    wndclass.cbClsExtra=0;
    wndclass.cbWndExtra=0;
    wndclass.lpfnWndProc=WndProc;
    wndclass.hIcon=LoadIcon(NULL,IDI_APPLICATION);
    wndclass.hCursor=LoadCursor(NULL,IDC_ARROW);
    wndclass.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH);
    wndclass.hInstance=hInstance;
    wndclass.lpszClassName=AppName;
    wndclass.lpszMenuName=NULL;
    wndclass.hIconSm=LoadIcon(NULL,IDI_APPLICATION);
    // register window class
    RegisterClassEx(&wndclass);
    // create window
    hwnd=CreateWindow(AppName,
                      TEXT("Client Of Exe Server"),
                      WS_OVERLAPPEDWINDOW,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      NULL,
                      NULL,
                      hInstance,
                      NULL);
    ShowWindow(hwnd,nCmdShow);
    UpdateWindow(hwnd);
    // message loop
    while(GetMessage(&msg,NULL,0,0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return(msg.wParam);
}
// Window Procedure
```

```
53  LRESULT CALLBACK WndProc(HWND hwnd,UINT iMsg,WPARAM wParam,LPARAM lParam)
54  {
55      // variable declarations
56      IDispatch *pIDispatch=NULL;
57      HRESULT hr;
58      DISPID dispid;
59      OLECHAR *szFunctionName=L"SumOfTwoIntegers";
60      VARIANT varg[2];
61      DISPPARAMS param={varg,0,2,NULL};
62      int n1,n2;
63      // code
64      switch(iMsg)
65      {
66      case WM_CREATE:
67          // initialize COM library
68          hr=CoInitialize(NULL);
69          if(FAILED(hr))
70          {
71              MessageBox(hwnd,TEXT("COM library can not be initialized"),TEXT("COM
                  Error"),MB_OK);
72              DestroyWindow(hwnd);
73              exit(0);
74          }
75          // get ISum Interface
76          hr=CoCreateInstance(CLSID_SumAutomation,
77                              NULL,
78                              CLSCTX_LOCAL_SERVER,
79                              IID_IDispatch,
80                              (void **)&pIDispatch);
81          if(FAILED(hr))
82          {
83              MessageBox(hwnd,TEXT("Component Can Not Be Created"),TEXT("COM
                  Error"),MB_OK|MB_ICONERROR|MB_TOPMOST);
84              DestroyWindow(hwnd);
85              exit(0);
86          }
87          hr=pIDispatch->GetIDsOfNames(IID_NULL,
88                                       &szFunctionName,
89                                       1,
90                                       GetUserDefaultLCID(),
91                                       &dispid);
92          if(FAILED(hr))
93          {
94              MessageBox(NULL,TEXT("Can Not Get ID For Function"),TEXT
                  ("Error"),MB_OK|MB_ICONERROR|MB_TOPMOST);
95              pIDispatch->Release();
96              DestroyWindow(hwnd);
97          }
98          n1=75;
99          n2=25;
100         // as DISPPARAMS rgvarg member receives parameters in reverse order
101         VariantInit(varg);
```

```
102         varg[0].vt=VT_INT;
103         varg[0].intVal=n2;
104         varg[1].vt=VT_INT;
105         varg[1].intVal=n1;
106         param.cArgs=2;
107         param.cNamedArgs=0;
108         param.rgdispidNamedArgs=NULL;
109         // reverse order of parameters
110         param.rgvarg=varg;
111         hr=pIDispatch->Invoke(dispid,
112                               IID_NULL,
113                               GetUserDefaultLCID(),
114                               DISPATCH_METHOD,
115                               &param,
116                               NULL,
117                               NULL,
118                               NULL);
119         if(FAILED(hr))
120         {
121             MessageBox(NULL,TEXT("Can Not Invoke Function"),TEXT("Error"),MB_OK|
                MB_ICONERROR|MB_TOPMOST);
122             pIDispatch->Release();
123             DestroyWindow(hwnd);
124         }
125         VariantClear(varg);
126         pIDispatch->Release();
127         DestroyWindow(hwnd);
128         break;
129     case WM_DESTROY:
130         CoUninitialize();
131         PostQuitMessage(0);
132         break;
133     }
134     return(DefWindowProc(hwnd,iMsg,wParam,lParam));
135 }
136
```