```c
#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      ISum *pIDispatch=NULL;
57      HRESULT hr;
58      DISPID dispid;
59      OLECHAR *szFunctionName=L"SumOfTwoIntegers";
60      VARIANT varg[2];
61      DISPPARAMS param={NULL,NULL,0,2};
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->lpVtbl->GetIDsOfNames(pIDispatch,
88                                               &IID_NULL,
89                                               &szFunctionName,
90                                               1,
91                                               GetUserDefaultLCID(),
92                                               &dispid);
93          if(FAILED(hr))
94          {
95              MessageBox(NULL,TEXT("Can Not Get ID For Function"),TEXT
                  ("Error"),MB_OK|MB_ICONERROR|MB_TOPMOST);
96              pIDispatch->lpVtbl->Release(pIDispatch);
97              DestroyWindow(hwnd);
98          }
99          n1=800;
100         n2=200;
101         // as DISPPARAMS rgvarg member receives parameters in reverse order
```

```
102         VariantInit(varg);
103         varg[0].vt=VT_INT;
104         varg[0].intVal=n2;
105         varg[1].vt=VT_INT;
106         varg[1].intVal=n1;
107         param.cArgs=2;
108         param.cNamedArgs=0;
109         param.rgdispidNamedArgs=NULL;
110         // reverse order of parameters
111         param.rgvarg=varg;
112         hr=pIDispatch->lpVtbl->Invoke(pIDispatch,
113                                       dispid,
114                                       &IID_NULL,
115                                       GetUserDefaultLCID(),
116                                       DISPATCH_METHOD,
117                                       &param,
118                                       NULL,
119                                       NULL,
120                                       NULL);
121         if(FAILED(hr))
122         {
123             MessageBox(NULL,TEXT("Can Not Invoke Function"),TEXT("Error"),MB_OK|
                MB_ICONERROR|MB_TOPMOST);
124             pIDispatch->lpVtbl->Release(pIDispatch);
125             DestroyWindow(hwnd);
126         }
127         VariantClear(varg);
128         pIDispatch->lpVtbl->Release(pIDispatch);
129         DestroyWindow(hwnd);
130         break;
131     case WM_DESTROY:
132         CoUninitialize();
133         PostQuitMessage(0);
134         break;
135     }
136     return(DefWindowProc(hwnd,iMsg,wParam,lParam));
137 }
138
```