DLL’s C# en PowerBuilder

Registrar la dll con el regsvr32 y desde PowerBuilder abrir el **Browser** ir a la solapa **OLE** y alli **programmable Objects** y buscar el nombre del complemento registrado. En **class information**, en la linea **Program ID**, lo que se indique será el nombre que debe utilizarse en la línea de código donde se hace la conexión al Objeto, por ejemplo:

Winsock.ConnectToNewObject("MSwinsock.Winsock.1")

luego si se abre la rama **functions,** deberían aparecer allí las funciones que implementa la dll.

existen casos en que estas funciones no se muestran, pero si la conexión se lleva a cabo sin error, allí debiera conociendo el nombre de las funciones, poder utilizarlas

Crea un Ole Object

//Declara Objeto

Long li\_rtn

OleObject Winsock

// Crear Objeto

Winsock = create OLEObject

//Conectar   
li\_rtn = Winsock.ConnectToNewObject("MSwinsock.Winsock.1")

if li\_rtn <> 0 then

MessageBox('Winsock','No puede Conectarse')

else

//Aqui el tratamiento de las funciones

Winsock.RemoteHost ("255.255.255.255")

destroy Winsock

Return 0

end if

[Use C# created DLL in PowerBuilder 12.5.2](http://stackoverflow.com/questions/34306591/use-c-sharp-created-dll-in-powerbuilder-12-5-2)

I have created a simple C# class library targetting .Net Framework 4.5.2 with one class using Visual Studio 2015 enterprise edition.

namespace PwdEncryptor {

public class Class1 {

public string Encrypt (string actualPassword){

return String.Concat(actualPassword, "Encrypt");

}

}

}

want to use this from my PowerBuilder code on another system. Purpose of doing this is to have common code for encryption of password.

I used it in my powerbuilder code by declaring in the Global instance variables like this:

Function string Encrypt(string actualPassword) Library "PwdEncryptor.dll"

and in Open() event of the application I wrote:

string pwd

pwd = Encrypt("XYZ")

When I ran the code, I got a message saying **"Unknown function name"**.

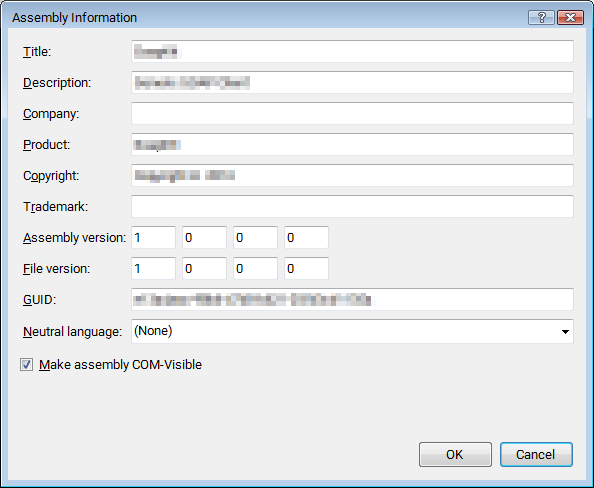
To overcome this I tried the solution mentioned [here](http://blogs.artinsoft.net/Mrojas/archive/2009/03/03/Calling-NET-from-PowerBuilder.aspx). A small deviation that I took was instead of doing the execution in powerbuilder on the same system I exported the registry which was created & imported to other system where powerbuilder code exists. In this case the error that I got was **Bad runtime function refernce at line in Open event of Application object.**

Is there a way that I can possibly use the DLL I have created? Am I missing something? Please advise.

Is the DLL registered on the system which you run your PowerBuilder code?

Yes, it is registered in the Registry. It gets registered as a part of the build on my system from the sample solution present in the link I mentioned in the question. FYI, it gets registered under HKEY\_CLASSES\_ROOT. I import this registry & then export it on the system having PowerBuilder code.

You have to expose DLL as COM visible. Detailed info at [this answer](http://stackoverflow.com/questions/842017/net-interop-in-powerbuilder?rq=1) and [this other](http://stackoverflow.com/questions/7092553/turn-a-simple-c-sharp-dll-into-a-com-interop-component).



Thanks for your response. I have already made the dll COM visible as suggested in the link I have provided in the question. Maybe you could have a look at the link & then advise if I'm doing anything wrong.

I have resolved this.

Instead of exporting the registry from my system & importing it on the system containing powerbuilder code, I should have registered my C# dll using **RegAsm.exe**

Once the DLL is registered, the PowerBuilder code is able to use the C# code.

Crear una DLL (COM) en Visual Studio C# para que sea visible desde Power Builder (PB)

Voy a explicar los pasos para poder **crear una DLL en C# usando Visual Studio**, y que dicha DLL pueda **ser usada desde otro lenguaje de programación**, en este caso **Power Builder (PB)**. Como digo, podrá ser utilizada desde cualquier otro lenguaje de programación, porque la DLL será creada como un componente **COM**.

Presupongo que el lector ya sabe crear, compilar y ejecutar un proyecto vacio y básico desde ambos lenguajes

**En Visual Studio**:

Creamos un nuevo proyecto C# para Windows de tipo Biblioteca de clases, y ponemos correctos (o si prefieres puedes dejar los nombres por defecto) el nombre de fichero y nombre de clase que establece Visual Studio por defecto.

Abrimos las **propiedades** de nuestro proyecto -> **Aplicación** -> **Información de ensamblado…** -> Marcar el checkbox de “**Crear ensamblado visible a través de COM**”

Volvemos de nuevo a la pantalla principal de las **propiedades**, ahora nos vamos a **Firma**, marcamos “**Firmar el ensamblado**” y nos **creamos una nueva firma**.

Dejamos el código fuente de la clase de la siguiente manera (en negrita marco lo más importante):

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Windows.Forms;  
**using System.Runtime.InteropServices;**

namespace Test  
{

**[Guid(“F8E03FC1-3367-4d4a-AD49-EF3EAEBFA6EC”)]**  
**[InterfaceType(ComInterfaceType.InterfaceIsDual)]**

public interface \_CTest

{

void Test();

void Test2();

}

**[Guid(“F8E03FC1-3367-4d4a-AD49-EF3EAEBFA6EC”)]**

**[ClassInterface(ClassInterfaceType.None)]**

**[ProgId(“Test.CTest”)]**

public class CTest : \_CTest

{

public void Test()

{

MessageBox.Show(“Hola, yo soy Test()”);

}

public void Tes2t()

{

MessageBox.Show(“Hola, yo soy Test2()”);

}

}

}

Los códigos de GUID se pueden obtener con alguna de las herramientas de Microsoft Visual Studio, o bien desde alguna web tipo <http://www.guidgen.com/>

Una vez hecho esto, compilamos el proyecto y nos aseguramos que no haya ningún error.

Lo más probable es que queramos distribuir la DLL en otros PCs, junto al programa realizado en Power Builder, para ello pasaremos a agregar/crear un **nuevo proyecto instalador** tipo “Otros tipos de proyectos” -> “Instalación e implementación” -> “Instalador de Visual Studio” -> “Proyecto de instalación”

En el nuevo proyecto instalador, agregaremos los resultados del proyecto de la clase de la DLL; y lo más importante, en las propiedades del resultado principal de dicha clase, especificaremos la propiedad **Register** a **vsdrpCOM**.

Generamos el nuevo proyecto instalador, y pasaríamos a instalar la DLL en el equipo.

Nota: Es posible que en Visual Studio, cuando generemos el instalador, se nos presente un mensaje tipo **Visual Studio registry capture utility has stopped working** o **Visual Studio registry capture utility dejo de funcionar**, para solucionarlo:

1. Localiza **regcap.exe** en: C:\Program Files\Microsoft Visual Studio 9.0\Common7\Tools\Deployment (o similar)
2. Botón derecho sobre el fichero y selecciona “**Propiedades**“.
3. Selecciona la pestaña de “**Compatibilidad**“
4. Marca el checkbox de **modo de compatibilidad** y selecciona **Windows Vista SP2**.
5. Marca el checkbox de **ejecutar como administrador**.

**En PowerBuilder:**

Simplemente instanciaremos en nuestro código fuente la DLL COM de la siguiente manera:

iole = CREATE OLEObject  
li\_rc = iole.ConnectToNewObject(“**Test.CTest**“)  
if li\_rc <> 0 then  
DESTROY iole  
end if

Y… ¡listo!