

Re: .NET and COM Interop

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.interop/2004-07/0369.html>

From: Nick Hall (nickh_at_aslan.nospam.co.uk)

Date: 07/23/04

Date: Fri, 23 Jul 2004 13:23:35 +0100

Comments inline: –

"Stephan Scheidl" <stephan.scheidl@gmx.at> wrote in message
news:40ffd6d2\$1@e-post.inode.at...

> *I have written a class library in c# and want to use it in VB6. Everything
> works fine so far but when I try to assign a value (an array of type
defined*

> *in c#) to a property defined on the c# side the vb compiler does not want
> to compile. Reading access to the property is working.
> I added a function that sets the array and it worked, so what is going
wrong*

> *in the property call?*

> *I extracted the IDL code from the type library and there is the problem:*

>

>

> [

> *uuid(D9284674-EBA7-4C9B-A823-65193E0D5FC3),*

> *version(1.0),*

> *custom(0F21F359-AB84-41E8-9A78-36D110E6D2F9, xxxxx.IColorRangeSelectEx)*

>]

> *dispinterface IColorRangeSelectEx {*

> *properties:*

> *methods:*

> [*id(0x60020000)*]

> *void SetColorSteps([in, out] SAFEARRAY(ColorStep)* Steps);*

> [*id(0x60020001)*]

> *SAFEARRAY(ColorStep) GetColorSteps();*

> [*id(0x60020002), propget*]

> *SAFEARRAY(ColorStep) ColorSteps();*

> [*id(0x60020002), propput*]

> *void ColorSteps([in] SAFEARRAY(ColorStep) rhs);*

>];

>

> *Take a look at the SetColorSteps and propput definitions. How can I solve*

> *this???*

>

I had the same problem a while back. It seems to be a limitation of the current marshalling engine. I posted a question on here a while back but no-one seemed to have any explanation. The only way I could get this to work as I wanted to was to do the following: –

1. Define the interface in a type library (with the extra level of indirection that VB needs on the propput);
2. Reference the created type library from my project;
3. To implement the interface properly you have to be a bit sneaky. If you try to implement the property properly, the compiler will complain. If, however, you define the get/set methods explicitly, the compiler is satisfied.

For example, I defined the following interface in a type library –

```
interface ISignature : IDispatch {
.....
[id(0x60020002), propget]
HRESULT SignatureBytes([out, retval] SAFEARRAY(unsigned char)* retVal);
[id(0x60020002), propput]
HRESULT SignatureBytes([in] SAFEARRAY(unsigned char)* retVal);
....
}
```

This comes though in DotNet as : –
`public System.Array SignatureBytes[get, set]`

Which I implemented like this in my class: –

```
Array ISignature.get_SignatureBytes()
{
return this.SignatureBytes;
}
void ISignature.set_SignatureBytes(ref Array value)
{
this.SignatureBytes = (Byte[]) value;
}
```

This works in the current version but a future version could stop this working. Hopefully, they'll make the proper fix (which is to allow arrays to be exposed to COM in a VB-friendly manner).

>
>
> *Another question:*
>
> *Everytime I change some code an recompile the dll I have to fix the*
> *reference in VB. I have assigned guids to all public types and I checked*
> *that the entries in the registry do not change. What am I doing wrong?*
>
>

>

> *Many thanks in advance*

>

>

>

> *Stephan Scheidl*

>

>

Have you assigned a GUID to the assembly (i.e. the type library)? I had this problem initially because there is a difference in behaviour between VB and C# – VB automatically creates an [assembly: Guid(...)] attribute in the AssemblyInfo file whereas C# does not. Otherwise, I think the type library GUID is generated automatically each time the assembly is compiled, causing COM clients to lose track.

Hope this helps,

Nick Hall