.Net WebServices and out/ref WebMethod arguments

Asked 16 years ago Modified 3 years, 10 months ago Viewed 16k times

12

I've received some documentation from one of our suppliers for a webservice they're publishing and they're very specific that on one of their WebMethods that an argument has the out modifier(? not sure if that's the right descriptor) for instance consider the following WebMethod signature:

```
[WebMethod]
public void HelloWorld(out string strVal)
{
    strVal = "Hello World";
}
```

[Obviously the actual method isn't a Hello World method]

Now, I'd never considered designing a WebMethod with an out/ref argument and it got me wondering why they would've used it.

Trying to understand an application for this design decision I threw a prototype together with a few basic Hello World style webmethods...one with a single out string argument, one with two out string arguments and one that doesn't receive any arguments but returns a string.

Upon trying to reference my webmethods from a separate application I notice that I have to access the method with the single out string argument exactly as if I'd defined the method to output the string so that in effect as far as the client is concerned:

```
public string HelloWorld1()
{
  return "Hello World";
}
```

and

```
public void HelloWorld2(out string strVal)
{
  strVal = "Hello World";
}
```

are exactly the same...in that I have to reference them both as such [where x is substituted for the correct method]:

```
string val = HelloWorldX();
```

Having attempted to reference the methods in the way I would access them if they weren't web methods [like so]:

```
string val = string.Empty;
MyService1.HelloWorld(out val);
Console.WriteLine(val);
```

which causes a compilation error stating that no method arguments accept 1 input. Why is that? There's obviously a web method that accepts one argument - I'm looking at it [HelloWorld2].

Upon examining the SOAP responses, I notice that the content of the response for HelloWorld1 is:

```
<HelloWorld1Response xmlns="http://tempuri.org/">
  <HelloWorld1Result>string</HelloWorld1Result>
</HelloWorld1Response>
```

And HelloWorld2 is

```
<HelloWorld2Response xmlns="http://tempuri.org/">
  <strVal>string</strVal>
</HelloWorld2Response>
```

Going a step further I thought, what if I have 2 ref arguments...

```
public void HelloWorld3(out string strVal1, out string strVal2)
{
    strVal1 = "Hello World";
    strVal2 = "Hello World Again!";
}
```

This generates the SOAP content:

```
<HelloWorld3Response xmlns="http://tempuri.org/">
  <strVal1>string</strVal1>
  <strVal2>string</strVal2>
</HelloWorld3Response>
```

I thought fair enough, so theoretically [providing I can figure out a way to pass out/ref arguments to WebMethods] that means I can just pass in two arguments that can be set by the method, but when I do this:

```
string val1 = string.Empty;
string val2 = string.Empty;
MyService1.HelloWorld3(out val1,out val2);
Console.WriteLine(val1);
Console.WriteLine(val2);
```

I should get the same compilation error I saw when I tried to reference the HelloWorld2 this way. With the obvious exception that it's complaining about 2 arguments instead of 1 [and in fact I do get the same exception, I tested it].

- What gives?
- Is there a reason or a way to use out/ref arguments in WebMethods that I'm missing?
- If there is, how do I reference WebMethods with multiple out/ref arguments?

c# .net vb.net web-services

Share Improve this question Follow

asked Dec 18, 2008 at 23:08
BenAlabaster
39.8k ● 22 ● 114 ● 151

2 Answers

Sorted by: Highest score (default)

17

I don't know what the protocol is for providing answers to your own questions, but the article referenced by Steven Behnke provided some clues for me to deduce a solution to this bizarre situation. And rather than leave everyone else to figure out what the implications are, I thought I share my findings.

So, consider the following webmethods defined in my WebService:

```
[WebMethod]
public string Method1()
{
    return "This is my return value";
}

[WebMethod]
public void Method2(out string strVal1)
{
    strVal1 = "This is my value passed as an output";
    //No return value
}

[WebMethod]
public void Method3(out string strVal1, out string strVal2)
```

```
{
    strVal1 = "This is my strVal1 value passed as an output";
    strVal2 = "This is my strVal2 value passed as an output";
    //No return value
}

[WebMethod]
public string Method4(out string strVal1, out string strVal2)
{
    strVal1 = "This is my strVal1 value passed as an output";
    strVal2 = "This is my strVal2 value passed as an output";
    return "This is my return value";
}
```

Now according to the document, the first parameter defined as Out, if the method returns void, then the first parameter is automatically used as the return parameter. So I would access each of my methods as follows:

Method1:

```
public string Method1() {}

var str = svc.Method1();
Console.WriteLine(str);
```

Method2:

```
public void Method2(out string strVal1) {}

var str = svc.Method2();
Console.WriteLine(str);
```

So you access them both in exactly the same manner, which is extremely confusing. Who on Earth would figure that out without having been told that by someone else? It is beyond my comprehension, how this could be a good idea.

Method3:

```
public void Method3(out string strVal1, out string strVal) {}

var str2 = String.Empty;
var str1 = svc.Method3(out str2);
Console.WriteLine(str1);
Console.WriteLine(str2);
```

Method4:

```
public string Method4(out string strVal1, out string strVal2) {}

var str1 = String.Empty;
var str2 = String.Empty;
var str3 = svc.Method4(out str1, out str2);
Console.WriteLine(str1);
Console.WriteLine(str2);
Console.WriteLine(str3);
```

So as you notice - if the method signature doesn't provide a return value [that is returns void], then the first param becomes the return value. If it already provides a return value, then it doesn't.

This can be extremely confusing for someone that hasn't come across that document. Many thanks for providing that link Steven - I really appreciate it.

To whomever decided that design pattern was a good idea to be written into the .NET Framework - I can't think what would've posessed you to think that was a good idea. I really dislike you quite intensely after all that.

ADDENDUM:

What I only just realised is that to add to the confusion, if you use *ref* instead of *out* then you *don't* do this, you'd treat the WebMethods exactly as you would have if you'd used them to call a regular method inside your application:

```
[WebMethod()]
public void Method3(ref string strVal1, ref string strVal2)
{
    strVal1 = "First argument return value";
    strVal2 = "Second argument return value";
}
```

Now to call that you'd use:

```
string val1 = String.Empty;
string val2 = String.Empty;
svc.Method3(ref val1, ref val2);
Console.WriteLine(val1);
Console.WriteLine(val2);
```

This inconsistency is mindboggling. The fact, that it is by design, is incomprehensible to me.

Share

Improve this answer

Follow

edited Feb 23, 2021 at 12:16

peterh
12.6k ● 20 ● 89 ● 113

answered Dec 19, 2008 at 4:30

BenAlabaster
39.8k ● 22 ● 114 ● 151

When you get to 2000 reputation, I believe the correct protocol would be to edit Steven Behnke's answer ... as intrusive as it may feel. But good job posting your summary. +1
– John MacIntyre Dec 19, 2008 at 14:17

2

Maybe this will help:

http://kbalertz.com/322624/Proxy-Class-First-Parameter-Service-Method-Returns-Return-Value-Reference.aspx

My favorite part is:

> STATUS
>
> This behavior is by design.

Share Improve this answer Follow

answered Dec 19, 2008 at 1:39

Steven Behnke
3,344 ● 3 ● 28 ● 35

Okay, thank you - that article was very informative. My only comment on that design idea is: "Who the hell came up with that idea, it's screwed!" – BenAlabaster Dec 19, 2008 at 2:53