Parameter Passing in C-Sharp

Parameter Passing in C#

For each parameter of a Slice operation, the C# mapping generates a corresponding parameter for the corresponding method in the `<interface-name>Operations_` interface. In addition, every operation has an additional, trailing parameter of type `Ice.Current`. For example, the `name` operation of the `Node` interface has no parameters, but the `name` method of the `NodeOperations_` interface has a single parameter of type `Ice.Current`. We will ignore this parameter for now.

Parameter passing on the server side follows the rules for the client side. To illustrate the rules, consider the following interface that passes string parameters in all possible directions:

Slice

```
module M {
    interface Example {
        string op(string sin, out string sout);
    };
};
```

The generated method for op looks as follows:

C#

```
public interface ExampleOperations_
{
    string op(string sin, out string sout, Ice.Current __current);
}
```

As you can see, there are no surprises here. For example, we could implement op as follows:

C#

```
using System;

public class ExampleI : ExampleDisp_
{
    public override string op(string sin, out string sout, Ice.Current current)
    {
        Console.WriteLine(sin);      // In params are initialized
        sout = "Hello World!";       // Assign out param
        return "Done";
    }
}
```

This code is in no way different from what you would normally write if you were to pass strings to and from a method; the fact that remote procedure calls are involved does not affect your code in any way. The same is true for parameters of other types, such as proxies, classes, or dictionaries: the parameter passing conventions follow normal C# rules and do not require special-purpose API calls.

See Also

- Server-Side C-Sharp Mapping for Interfaces
- Raising Exceptions in C-Sharp
- Tie Classes in C-Sharp
- The Current Object