Serializable Objects in C-Sharp

In addition to serializing Slice types, applications may also need to incorporate foreign types into their Slice definitions. Ice allows you to pass CLR serializable objects directly as operation parameters or as fields of another data type. For example:

Slice

```
["clr:serializable:SomeNamespace.CLRClass"]
sequence<byte> CLRObj;
struct MyStruct {
    int i;
    CLRObj o;
};

interface Example {
    void op(CLRObj o, MyStruct s);
};
```

The generated code for `MyStruct` contains member `i` of type `int` and a member `o` of type `SomeNamespace.CLRClass`:

C#

```
public partial class MyStruct : _System.ICloneable {
    public int i;
    SomeNamespace.CLRClass o;

    // ...
}
```

Similarly, the signature for `op` has parameters of type `CLRClass` and `MyStruct`:

C#

```
void op(SomeNamespace.CLRClass o, MyStruct s);
```

Of course, your client and server code must have an implementation of `CLRClass` that sets the `Serializable` attribute:

C#

```
namespace SomeNamespace {
    [Serializable]
    public class CLRClass {
        // ...
    }
}
```

You can implement this class in any way you see fit — the Ice run time does not place any other requirements on the implementation. However, note that the CLR requires the class to reside in the same assembly for client and server.

See Also

- Serializable Objects