Stream Helper Methods in C-Sharp

Previous Next

The stream classes provide all of the low-level methods necessary for encoding and decoding Ice types. However, it would be tedious and error-prone to manually encode complex Ice types such as classes, structs, and dictionaries using these low-level functions. For this reason, the Slice compiler optionally generates helper methods for streaming complex Ice types.

We will use the following Slice definitions to demonstrate the language mapping:

Slice

```
module M
{
    sequence<...> Seq;
    dictionary<...> Dict;
    struct S
    {
        ...
    }
    enum E { ... }
    class C
    {
        ...
    }
    interface I
    {
        ...
    }
}
```

The Slice compiler generates the corresponding helper methods shown below:

C#

```
namespace M
{
    public sealed class SeqHelper
    {
        public static ...[] read(Ice.InputStream istr);
        public static void write(Ice.OutputStream ostr, ...[] v);
    }

    public sealed class DictHelper
    {
        public static Dictionary<...> read(Ice.InputStream istr);
        public static void write(Ice.OutputStream ostr, Dictionary<...> v);
    }

    public partial struct S
    {
        public static S ice_read(Ice.InputStream istr);
        public static void ice_write(Ice.OutputStream ostr, S v);
        ...
        // Instance methods
        public void ice_read(Ice.InputStream istr);
        public void ice_write(Ice.OutputStream ostr);
    }

    public sealed class EHelper
    {
        public static E read(Ice.InputStream istr);
        public static void write(Ice.OutputStream ostr, E v);
    }

    public sealed class IPrxHelper : ...
    {
        public static IPrx read(Ice.InputStream istr);
        public static void write(Ice.OutputStream ostr, IPrx v);
        ...
    }
}
```

The `IPrxHelper` class provides `read` and `write` methods for extracting and inserting proxies, respectively. Note that the `read` method returns a proxy of type `IPrx` but does not perform the equivalent of a `checkedCast` to verify that the remote object supports interface `I`.

No additional code is generated for marshaling instances of class types, such as type `C` that we defined above. Applications should use `OutputStream.writeValue` and `InputStream.readValue` to insert and extract class instances, respectively.

Back to Top ^

See Also

- slice2cs Command-Line Options
- The InputStream Interface in C-Sharp
- The OutputStream Interface in C-Sharp

Previous Next