Simple Example of Class Encoding

To make the preceding discussion more concrete, consider the following class definitions:

Slice

```
interface SomeInterface {
     void op1();
};

class Base {
    int baseInt;
    void op2();
    string baseString;
};

class Derived extends Base implements SomeInterface {
    bool derivedBool;
    string derivedString;
    void op3();
    double derivedDouble;
};
```

Note that `Base` and `Derived` have operations, and that `Derived` also implements the interface `SomeInterface`. Because marshaling of classes is concerned with state, not behavior, the operations `op1`, `op2`, and `op3` are simply ignored during marshaling and the on-the-wire representation is as if the classes had been defined as follows:

Slice

```
class Base {
    int baseInt;
    string baseString;
};

class Derived extends Base {
    bool derivedBool;
    string derivedString;
    double derivedDouble;
};
```

Suppose the sender marshals two instances of `Derived` (for example, as two in-parameters in the same request) with these member values:

First instance:

Member	Type	Value	Marshaled Size (in bytes)
`baseInt`	`int`	`99`	4
`baseString`	`string`	`"Hello"`	6
`derivedBool`	`bool`	`true`	1
`derivedString`	`string`	`"World!"`	7
`derivedDouble`	`double`	`3.14`	8

Second instance:

Member	Type	Value	Marshaled Size (in bytes)
`baseInt`	`int`	`115`	4
`baseString`	`string`	`"Cave"`	5
`derivedBool`	`bool`	`false`	1

derivedString	string	"Canem"	6
derivedDouble	double	6.32	8

The sender arbitrarily assigns a non-zero identity to each instance. Typically, the sender will simply consecutively number the instances starting at 1. For this example, assume that the two instances have the identities 1 and 2. The marshaled representation for the two instances (assuming that they are marshaled immediately following each other) is shown below:

Marshaled Value	Size in Bytes	Type	Byte offset
1 *(identity)*	4	int	0
0 *(marker for class type ID)*	1	bool	4
"::Derived" *(class type ID)*	10	string	5
20 *(byte count for slice)*	4	int	15
1 (*derivedBool*)	1	bool	19
"World!" (*derivedString*)	7	string	20
3.14 (*derivedDouble*)	8	double	27
0 *(marker for class type ID)*	1	bool	35
"::Base" *(type ID)*	7	string	36
14 *(byte count for slice)*	4	int	43
99 (*baseInt*)	4	int	47
"Hello" (*baseString*)	6	string	51
0 *(marker for class type ID)*	1	bool	57
"::Ice::Object" *(class type ID)*	14	string	58
5 *(byte count for slice)*	4	int	72
0 *(number of dictionary entries)*	1	size	76
2 *(identity)*	4	int	77
1 *(marker for class type ID)*	1	bool	81
1 *(class type ID)*	1	size	82
19 *(byte count for slice)*	4	int	83
0 (*derivedBool*)	1	bool	87
"Canem" (*derivedString*)	6	string	88
6.32 (*derivedDouble*)	8	double	94
1 *(marker for class type ID)*	1	bool	102
2 *(class type ID)*	1	size	103
13 *(byte count for slice)*	4	int	104
115 (*baseInt*)	4	int	108
"Cave" (*baseString*)	5	string	112
1 *(marker for class type ID)*	1	bool	117
3 *(class type ID)*	1	size	118
5 *(byte count for slice)*	4	int	119
0 *(number of dictionary entries)*	1	size	123

Note that, because classes (like exceptions) are sent as a sequence of slices, the receiver of a class can slice off any derived parts of a class it does not understand. Also note that (as shown in the above table) each class instance contains three slices. The third slice is for the type `::Ice::Object`, which is the base type of all classes. The class type ID `::Ice::Object` has the number 3 in this example because it is the third distinct type ID that is marshaled by the sender. (See entries at byte offsets 58 and 118 in the above table.) All class instances have this final slice of type `::Ice::Object`.

Marshaling a separate slice for `::Ice::Object` dates back to Ice versions 1.3 and earlier. In those versions, classes carried a facet map that was marshaled as if it were defined as follows:

Slice

```
module Ice {
    class Object;

    dictionary<string, Object> FacetMap;

    class Object {
        FacetMap facets; // No longer exists
    };
};
```

As of Ice version 1.4, this facet map is always empty, that is, the count of entries for the dictionary that is marshaled in the `::Ice::Object` slice is always zero. If a receiver receives a class instance with a non-empty facet map, it must throw a `MarshalException`.

Note that if a class has no data members, a type ID and slice for that class is still marshaled. The byte count of the slice will be 4 in this case, indicating that the slice contains no data.

See Also

- Data Encoding for Classes
- Data Encoding for Exceptions
- Basic Data Encoding
- Type IDs