Data Encoding for Interfaces

Interfaces can be marshaled by value. For an interface marshaled by value (as opposed to a class instance derived from that interface), only the type ID of the most-derived interface is encoded. Here are the Slice definitions once more:

Slice

```
interface Base { /* ... */ };

interface Derived extends Base { /* ... */ };

interface Example {
    void doSomething(Base b);
};
```

If the client passes a class instance to `doSomething` that does not have a Slice definition (but derives from `Derived`), the on-the-wire representation of the interface is as follows:

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
`4` *(byte count for slice)*	4	`int`	15
`0` *(marker for class type ID)*	1	`bool`	19
`"::Ice::Object"` *(class type ID)*	14	`string`	20
`5` *(byte count for slice)*	4	`int`	34
`0` *(number of dictionary entries)*	1	`size`	38

See Also

- Passing Interfaces by Value
- Type IDs
- Data Encoding for Classes