Objective-C Mapping for Interfaces by Value

Slice permits you to pass an interface by value:

Slice

```
interface ClassBase {
    void someOp();
    // ...
};

interface Processor {
    ClassBase process(ClassBase b);
};

class SomeClass implements ClassBase {
    // ...
};
```

Note that `process` accepts and returns a value of type `ClassBase`. This is *not* the same as passing `id<ClassBasePrx>`, which is a *proxy* to an object of type `ClassBase` that is possibly remote. Instead, what is passed here is an *interface*, and the interface is passed by *value*.

The immediate question is "what does this mean?" After all, interfaces are abstract and, therefore, it is impossible to pass an interface by value. The answer is that, while an interface cannot be passed, what *can* be passed is a class that implements the interface. That class is type-compatible with the formal parameter type and, therefore, can be passed by value. In the preceding example, `SomeClass` implements `ClassBase` and, hence, can be passed to and returned from the `process` operation.

The Objective-C mapping maps interface-by-value parameters to `ICEObject*`, regardless of the type of the interface. For example, the proxy protocol for the `process` operation is:

Objective-C

```
-(ICEObject *) process:(ICEObject *)b;
```

This means that you can pass a class of any type to the operation, even if it is not type-compatible with the formal parameter type, because all classes derive from `ICEObject`. However, an invocation of `process` is still type-safe at run time: the Ice run time verifies that the class instance that is passed implements the specified interface; if not, the invocation throws an `ICEMarshalException`.

Passing interfaces by value as `ICEObject*` is a consequence of the decision to not generate a formal protocol for classes. (If such a protocol would exist, the formal parameter type could be `id<`*`ProtocolName`*`>`. However, as we described for the class mapping, a protocol would require the implementation of a class to implement all of its operations, which can be inconvenient. Because it is rare to pass interfaces by value (more often, the formal parameter type will be a base *class* instead of a base *interface*), the minor loss of static type safety is an acceptable trade-off.

See Also

- Passing Interfaces by Value
- Objective-C Mapping for Classes