Subject: object suggestion for future IDL versions
Posted by John Persing on Mon, 21 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

I don't know if anybody else has had this problem, but it occurs with objects that take advantage of inheritance. The object may take advantage of the inherited methods by using the notation

self->method_name, arg1, arg2, etc

But if the new object redefines any of the inherited methods (for example "assign"), it cannot make reference to the inherited method. So say our new object is just like the inherited object, except that when the data is changed, a time stamp is placed on the data so

```
PRO dat_new__define
struct = {dat_new, time_stamp:'', INHERITS dat_old}
END
```

can define our new object. Ideally, I think, we would want to extend the assign method like this

```
PRO dat_new::assign, field_string, value
self->assign, field_string, value
self.time_string = STRING(SYSTIME())
END
```

I am not exactly sure if this is the way objects _should_ work, but this is the way I _want_ them to work. The work around is to put an extra level of abstraction in the definitions of the parent methods, or provide the new object with a new method name. The former solution leads to a duplication of every method, and more dangerously, two ways to do everything. The latter solution presents the even more dangerous possibility that the parent object method is available to edit the data without updating the time stamp. So I have settled on the solution of cut and pasting the parent definition of the method and making the one-line change that I need to in the new method.

--
}3 John Persing }3
http://www.frii.com/~persing persing@frii.com
Half of all Americans earn less than the median income!!!!!!