Subject: Re: Does this make sense? (scalar objects)
Posted by JD Smith on Wed, 03 Dec 2003 22:25:26 GMT
View Forum Message <> Reply to Message

On Wed, 03 Dec 2003 07:33:20 -0700, Marc Schellens wrote:

```
>  check this out:
>
>  file tt.pro:
>  pro o::test
>
>  help,self[[0]]
>  help,(self[[0]])
>
>  print,self[[0]].a
>  print,(self[[0]]).a ;; ???
>  end
>
>  pro tt
>
>  s={o,a:0}
>
>  print,s[[0]].a
>  print,(s[[0]]).a
>
>  obj=obj_new('o')
>
>  obj->test
>  end
>
>
>  IDL> tt
>  % Compiled module: TT.
>         0
>         0
>  <Expression>    OBJREF    = Array[1]
>  <Expression>    OBJREF    = Array[1]
>         0
>  % Object reference must be scalar in this context: <OBJREF    Array[1]>
>  % Execution halted at: O::TEST            7 /home/marc/idl/tt.pro %
>                  TT              19 /home/marc/idl/tt.pro %
>           $MAIN$
>
>
>  Doesn't make sense, does it?
```

Well, given that self is always a scalar, your attempts to index it are confusing. In any case, the notation a[[b]] creates a single element

vector:

```
IDL> a=1
IDL> print,size(a[[0]],/DIMENSIONS)
         1
```

You cannot do anything to more than one object at a time (e.g. no objarr method calls or instance variable dereference). Hence the error. The reason why self[[0]].a works, is that there is probably special code to handle instance variable derefence for a single element vector, which does not or cannot operate with (self[[0]]).a. Method calls don't like a vector no matter what: try

```
obj[[0]]->test
```

Confusing issues like this have lead at least one RSI programmer to long for the abolishment of the scalar as a separate type from a single element vector. Sadly, the chance to do this without breaking lots of code has long passed.

JD
