Subject: Re: Does this make sense? (scalar objects)
Posted by JD Smith on Fri, 05 Dec 2003 17:55:52 GMT
View Forum Message <> Reply to Message

On Fri, 05 Dec 2003 02:58:05 -0700, Marc Schellens wrote:

>> I cannot guess any example about which (IDL) code would be broken, if
>> single element vectors and scalars would be treated the same. Do you
>> have an example?
>> Or did you mean binary code linked to IDL?
>
> Sorry, please forget. I read your reply not careful enough. As you were
> talkin gabout the abolishment of scalar type, of course you are right.
> Nevertheless, apart from indexing there should not be any difference in
> behaviour.

For objects, it's quite clear why you can't apply methods across a vector of object variables:

```
IDL> objs=[obj_new('IDL_Container'), obj_new('MyFooObj')]
IDL> objs->DoSomeMethod ; WRONG
```

Since objects are generic pointers, and a vectors of objects can contain any combination of object classes, it's clear why you can't use this notation. The same is true of pointer arrays, for nearly the same reasons:

```
IDL> ptrs=[ptr_new('string'),ptr_new(indgen(5))]
IDL> print,*ptrs+5 ;WRONG
```

Single element vectors are different than scalars in several ways: they can be transposed, reformed, and rebinned, whereas scalars cannot, and they can have matrix multiplications applied to them, etc. A better way of asking the question is "What can't you do with scalars that you can do with vectors?". The answer to this consists of the long list of IDL vector operations discussed here daily. There may not be any *useful* distinctions between scalars and single-element vectors, but there are certainly plenty of programmatic distinctions, which would break backward compatibility if ignored --- hence, we are stuck with both.

JD