
Subject: Copying IDL objects

Posted by [rivers](#) on Wed, 30 Sep 1998 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

I am trying to figure out how to make a copy of an instance of an IDL object.

```
a = obj_new('myobj')
```

```
b = a
```

; This does not do what I want, it makes a new reference to the same instance

```
b = obj_copy(a) ; This is what I want
```

The only way I can see how to do it is:

```
pro my_obj::copy, out
```

```
  out->set_a, self.a
```

```
  out->set_b, self.b
```

```
  ....
```

```
end
```

This means I have to write a set_X function for each field, and each object class needs its own version of the ::copy routine. This seems crazy.

I seem to recall a post from a few months ago about some ways of generally getting/setting properties in IDL objects. Any ideas?

Mark Rivers	(773) 702-2279 (office)
Argonne National Laboratory	(630) 252-0422 (office)
Building 434A	(630) 252-0405 (lab)
9700 South Cass Avenue	(630) 252-1713 (beamline)
Argonne, IL 60439	(630) 252-0443 (FAX)

Subject: Re: Copying IDL objects

Posted by [J.D. Smith](#) on Fri, 02 Oct 1998 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Mark Rivers wrote:

>

> I am trying to figure out how to make a copy of an instance of an IDL object.

>

> a = obj_new('myobj')

> b = a

> ; This does not do what I want, it makes a new reference to the same instance

>

> b = obj_copy(a) ; This is what I want

>

> The only way I can see how to do it is:
>
> pro my_obj::copy, out
> out->set_a, self.a
> out->set_b, self.b
>
> end
>
> This means I have to write a set_X function for each field, and each object
> class needs its own version of the ::copy routine. This seems crazy.
>
> I seem to recall a post from a few months ago about some ways of generally
> getting/setting properties in IDL objects. Any ideas?
>

Here's something really dirty that will do the trick. Use it if
desperate:

```
function obj_copy, obj
  o=obj
  file=filepath('obj.sav',/TMP)
  save,o,FILENAME=file
  restore, FILENAME=file
  return,o
end
```

Does the job though. If you're really doing a lot of this, consider
setting up a small ram disk just for this purpose.

JD

--

J.D. Smith [*] WORK: (607) 255-5842
Cornell University Dept. of Astronomy [*] (607) 255-6263
304 Space Sciences Bldg. [*] FAX: (607) 255-5875
Ithaca, NY 14853 [*]

Subject: Re: Copying IDL objects
Posted by [davidf](#) on Fri, 02 Oct 1998 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi Folks,

I wrote just a couple of minutes ago:

> This seems like a lot of work. I think I would write this
> function something like this:

```
>
> PRO My_Obj::Copy, out
> tags = N_Tags(self)
> For j=0,tags-1 DO out.(j) = self.(j)
> END
```

Mark Rivers kindly points out to me that this doesn't work, for the simple reason that self is not a structure (even though it can be accessed as if it were a structure in a method). Thus, N_Tags returns a 0.

To make it work, I have to do this (with a bit of error checking too :-):

```
PRO My_Obj::Copy, out
selfClass = Obj_Class(self)
outClass = Obj_Class(out)
IF selfClass NE outClass THEN BEGIN
ok = Dialog_Message('Object classes not identical.')
RETURN
END
ok = Execute('struct = {' + selfClass + '}')
tags = N_Tags(struct)
For j=0,tags-1 DO out.(j) = self.(j)
END
```

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438, Toll-Free Book Orders: 1-888-461-0155
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Subject: Re: Copying IDL objects
Posted by [davidf](#) on Fri, 02 Oct 1998 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Mark Rivers (rivers@cars3.uchicago.edu) writes:

```
> I am trying to figure out how to make a copy of an instance of an IDL object.
>
> a = obj_new('myobj')
> b = a
```

```
> ; This does not do what I want, it makes a new reference to the same instance
>
> b = obj_copy(a) ; This is what I want
```

I thought Replicate might do it, but I see it also just replicates the object reference, not the actual object.

```
> The only way I can see how to do it is:
>
> pro my_obj::copy, out
>   out->set_a, self.a
>   out->set_b, self.b
>   ....
> end
>
> This means I have to write a set_X function for each field, and each object
> class needs its own version of the ::copy routine. This seems crazy.
```

This seems like a lot of work. I think I would write this function something like this:

```
PRO My_Obj::Copy, out
tags = N_Tags(self)
For j=0,tags-1 DO out.(j) = self.(j)
END
```

The only drawback is that N_Tags (and Tag_Names, too) will only find the tags or fields in the outermost structure. I do have a little recursive function on my web page to find *all* the tags in an embedded structure definition.

http://www.dfanning.com/tips/recursive_struct_tags.html

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438, Toll-Free Book Orders: 1-888-461-0155
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>
