Subject: structure arguments sometimes behave like value types - why?
Posted by justspam03 on Fri, 20 Apr 2007 20:10:29 GMT
View Forum Message <> Reply to Message

Hi all,

according to the IDL reference, structures (as a whole) are treated
like reference types when supplied as an argument. A small test
program of the kind

```
pro main
 ns = { nullableString, hasValue: 0, value: '' }
 a = { nullableString }
 a.value = '3'
 changeval, a
 print, a.value
end
pro changeval, s
 s.value = '4'
end
```

confirms this - after the call to changeval, a.value has value '4'.
However, when the structure is an object variable as in the example
appended below, it seems that only a copy of structtest.val is
exchanged, not a reference to it. The final call to printValue in
'main' prints a '1'.
Could someone please explain why? What am I missing?
Thanks a lot
 Oliver

```
pro main

 ns = { nullableString, hasValue: 0, value: '' }

 x = obj_new('structtest')   ; object contains membervariable of type
nullableString
 x->set, '1'                 ; set value of nullableString to '1'
 x->printValue               ; and print
 b = x->getStruct()          ; get reference? value?
 b.value = '2'
 x->printValue    ; result is still '1' ! Why?
end

function structtest::init
 return, 1
end
```

```
pro structtest::set, value
 self.val.value = value
end

pro structtest::printValue
 print, self.val.value
end

function structtest::getStruct
 return, self.val
end

pro structtest__define
 obj = { STRUCTTEST , val: {nullableString}   }
end
```
