Subject: Re: New Object Method Invocation Syntax Brokenness
Posted by JDS on Tue, 17 May 2011 16:17:46 GMT
View Forum Message <> Reply to Message

In my version of IDL 8.0, your example follows, but my example does not. I.e. self->item([1,2,3,4]) still correctly calls the method in IDL 8.0, but no longer does it in 8.0.1. My take is that IDL 8 introduced the syntax ambiguity ("->" and "." interchangeable), then IDL 8.0.1 reversed the precedence in ambiguous cases, now favoring structure/class variable dereferencing over method calling.

BTW, the documentation mentions this interchangeability in the context of method invocation:

  "Beginning with IDL 8.0, you can use the . and -> forms of the operator interchangeably; they are

   equivalent."

In a sense, this bug cannot be fixed, since it is inherent in the choice to make "." mean two things. Unless idl2 is in force, you simply cannot know what I mean by:

```
  d=a.b(c)
```

even (in the case of "b" being both a method name and a class variable), at runtime! ITT could certainly patch "->" to avoid breaking old code, but new code will always have this potential for silent brokenness (unless people shun "."). What's interesting is the main concern was putting off new users with meaningless syntax error messages. This example shows a much more problematic issue arises.

One possible fix would be to make array subscripting usage following the "." operator implicitly require brackets "[]". This would break old code like c=a.b(4), but leave intact other uses of parentheses for array indexing. I'd call this a "partial idl2 requirement".   It still leaves more than a year's worth of IDL versions in use silently breaking old code.

Just to make it ridiculously obvious to everyone:

```
;+++++++++++++++++++++++++
;  failure__define.pro
pro Failure::Explode
  prevent=[1b]  ;; Alway prevent the explosion, no matter what!
  if self->Prevent_Explosion(prevent) then $
     print, 'Explosion averted, go in peace.' $
  else print, '!BOOM! Nuclear war initiated.'
end

function Failure::Prevent_Explosion, confirm
  return, keyword_set(confirm[0])
end

pro failure__define
```

```
  st={FAILURE,$
      prevent_explosion: 0b}
end
;++++++++++++++++++++++++
```

```
IDL v5 - v7:
IDL> f=obj_new('failure')
IDL> f->Explode
Explosion averted, go in peace.
```

```
IDL >v8:
IDL> f=obj_new('failure')
IDL> f->Explode
!BOOM! Nuclear war initiated.
```

JD