Subject: Re: Object Method validity
Posted by Mark Hadfield on Mon, 11 Aug 2003 22:08:37 GMT
View Forum Message <> Reply to Message

Robert Moss wrote:
> Is there a way to verify the existance of a particular object method?
> Something like
>
> valid = Method_Valid(theObject, "HopedForMethod")
>
> I guess. I've spent a little while rtfm-ing and googling to no avail.

I believe that the only generally valid approach is to call the method and trap any errors with a CATCH statement. This has the advantage(??) of actually calling the method and checking that your parameters are valid. Something like this (Warning: I haven't checked it and haven't generalised it to handled function-type methods)...

```
  catch, err
  if err ne 0 then begin
    valid = 0B
    goto, finished
  endif
  call_method, theObject, 'HopedForMethod', param0
  valid = 1B
  catch, /CANCEL
  finished:
```

But be aware that if HopedForMethod is not bound to the class that theObject belongs to, then IDL will spend a significant amount of time searching the path for a file with the name

```
  obj_class(theObject)+'__hopedformethod.pro'
```

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Object Method validity
Posted by rmmoss on Tue, 12 Aug 2003 13:34:58 GMT
View Forum Message <> Reply to Message

Mark Hadfield <m.hadfield@niwa.co.nz> wrote in message news:<bh9457$ctm$1@newsreader.mailgate.org>...
> Robert Moss wrote:

```
>>  Is there a way to verify the existance of a particular object method?
>>  Something like
>>
>>  valid = Method_Valid( theObject, "HopedForMethod" )
>>
>>  I guess. I've spent a little while rtfm-ing and googling to no avail.
>
>  I believe that the only generally valid approach is to call the method
>  and trap any errors with a CATCH statement. This has the advantage(??)
>  of actually calling the method and checking that your parameters are
>  valid. Something like this (Warning: I haven't checked it and haven't
>  generalised it to handled function-type methods)...
>
>     catch, err
>     if err ne 0 then begin
>       valid = 0B
>       goto, finished
>     endif
>     call_method, theObject, 'HopedForMethod', param0
>     valid = 1B
>     catch, /CANCEL
>     finished:
>
>  But be aware that if HopedForMethod is not bound to the class that
>  theObject belongs to, then IDL will spend a significant amount of time
>  searching the path for a file with the name
>
>     obj_class(theObject)+'__hopedformethod.pro'
```

That's what I was afraid of. Oh well, time for another minor feature
request. Thanks for taking the time to answer.

Robert

Subject: Re: Object Method validity
Posted by Pavel Romashkin on Tue, 12 Aug 2003 22:10:40 GMT

View Forum Message <> Reply to Message

If IDL didn't check for a .pro file for the method then feature you want
would only check for *compiled* methods. Does not seem reasonable
because having a method in a separate .pro file is not uncommon.
Besides,

> IDL will spend a significant amount of time
> searching the path for a file with the name

is an overstatement. I am not sure what file system one must have for
this to become a problem.
I would stick with error catching.
Pavel

Robert Moss wrote:
>
>> But be aware that if HopedForMethod is not bound to the class that
>> theObject belongs to, then IDL will spend a significant amount of time
>> searching the path for a file with the name
>>
>>   obj_class(theObject)+'__hopedformethod.pro'
>
> That's what I was afraid of. Oh well, time for another minor feature
> request. Thanks for taking the time to answer.
>
> Robert

Subject: Re: Object Method validity
Posted by Mark Hadfield on Tue, 12 Aug 2003 22:15:25 GMT
View Forum Message <> Reply to Message

Robert Moss wrote:

 > Mark Hadfield <m.hadfield@niwa.co.nz> wrote in message
news:<bh9457$ctm$1@newsreader.mailgate.org>...
 >
 >>Robert Moss wrote:
 >>
 >>>Is there a way to verify the existance of a particular object method?
 >>>Something like
 >>>
 >>>valid = Method_Valid(theObject, "HopedForMethod")
 >>>
 >>>I guess. I've spent a little while rtfm-ing and googling to no avail.
 >>
 >>I believe that the only generally valid approach is to call the method
 >>and trap any errors with a CATCH statement. This has the advantage(??)
 >>of actually calling the method and checking that your parameters are
 >>valid. Something like this (Warning: I haven't checked it and haven't
 >>generalised it to handled function-type methods)...
 >>
 >> catch, err
 >> if err ne 0 then begin
 >>   valid = 0B
 >>   goto, finished

```
>>   endif
>>   call_method, theObject, 'HopedForMethod', param0
>>   valid = 1B
>>   catch, /CANCEL
>>   finished:
>>
>>But be aware that if HopedForMethod is not bound to the class that
>>theObject belongs to, then IDL will spend a significant amount of time
>>searching the path for a file with the name
>>
>>   obj_class(theObject)+'__hopedformethod.pro'
>
> That's what I was afraid of.  Oh well, time for another minor feature
> request.  Thanks for taking the time to answer.
```

Are you really trying to answer the question "Does this method exist?"
rather than "Will this method call work?"

If the latter, then I believe the "try it and see" approach is
appealing because it answers the question in the most direct way
possible. (I guess its major drawback is that it may be difficult to
clean up after the method call if it fails.) In this I am influenced
by my exposure to the Python language where "try it and see" is almost
always the preferred approach. ("Can I read another line from this
file?" "Try it and see!" "Will my database query work?" "Try it and
see!" "Does this object support the functionality I want?" "Try it &
see!")

If you really do want to answer the question "Does this method exist?"
then you *could* try to duplicate IDL's rules for resolving
methods. These are hinted at in the documentation, but there are
subtleties. I suggest you do a Google Groups search for a thread in
June 1998 entitled "Important object lesson". Here is an excerpt from
a posting of mine in that thread:

  This is my current working hypothesis, based on a little
  experimentation, a modest amoung of logic, generous conjecture & a
  minimal scanning of the documentation...

  If IDL encounters

    MyClass->MyMethod

  the three situations are:

    1. IDL finds a MyClass::Method in memory and uses it. (In the
    normal course of events the method will have been included in the
    myclass__define.pro, before the myclass__define procedure, so it

will have been compiled the first time an instance of the class
was created.) If MyClass::MyMethod is recompiled, the
modifications are recognised.

2. Not finding MyClass::Method, IDL searches up the inheritance
tree, finds a ASuperClass::Method in memory and uses it for the
remainder of the session. If MyClass::MyMethod is recompiled, the
modifications are not recognised, because this method is never
called. Which is confusing.

3. Failing 1 & 2, IDL searches the !path for myclass__mymethod.pro
(and maybe then for similar files for all superclasses). This can
take a while. For ordinary methods, failure to find it results in
an error. Obj_new and obj_destroy look for an Init and Cleanup
respectively, but if they fail to find them, they just skip that
step--until the next time & the next time & ...etc.

To look for methods that have already been compiled (steps 1 & 2), you
can use

```
   HELP, /ROUTINES, OUTPUT=routine_list
```

and then search routine_list for HopedForMethod, following the same
inheritance rules as IDL. Step 3 requires you to search the !PATH for
.pro and .sav files, but if you know that methods are always stored
along with class structure definitions in *__define.pro files then you
can skip this.

And then there's built-in methods and DLMs...Hmmm.

But, as I said above, if the question is really "Will this method call
work?" then "Try it and see!" is appealing

BTW I think my code, quoted above, was incorrect and should be

```
    catch, err
    if err ne 0 then begin
      valid = 0B
      goto, finished
    endif
    call_method, theObject, 'HopedForMethod', param0
    valid = 1B
    finished:
    catch, /CANCEL
```

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz

Subject: Re: Object Method validity
Posted by Craig Markwardt on Wed, 13 Aug 2003 17:00:48 GMT
View Forum Message <> Reply to Message

rmmoss@cox.net (Robert Moss) writes:
> Is there a way to verify the existance of a particular object method?
> Something like
>
> valid = Method_Valid(theObject, "HopedForMethod")
>
> I guess. I've spent a little while rtfm-ing and googling to no avail.

How about using:
```
  pronames = [routine_info(), routine_info(/func)]
  wh = where(pronames EQ strupcase(CLASSNAME+'::'+METHODNAME), ct)
  if ct GT 0 then print, 'Method found!' else print, 'Method NOT found!'
```

This assumes the class is already compiled. Also, if you want
system-level classes, you will need to use the appropriate keywords to
ROUTINE_NAMES().

Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: Object Method validity
Posted by Mark Hadfield on Wed, 13 Aug 2003 21:37:33 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
> rmmoss@cox.net (Robert Moss) writes:
>
>> Is there a way to verify the existance of a particular object method?
>> Something like
>>
>> valid = Method_Valid(theObject, "HopedForMethod")
>>
>> I guess. I've spent a little while rtfm-ing and googling to no avail.

>
>
> How about using:
> pronames = [routine_info(), routine_info(/func)]
> wh = where(pronames EQ strupcase(CLASSNAME+'::'+METHODNAME), ct)
> if ct GT 0 then print, 'Method found!' else print, 'Method NOT found!'
>
> This assumes the class is already compiled. Also, if you want
> system-level classes, you will need to use the appropriate keywords to
> ROUTINE_NAMES().

Good suggestion, but you also need to search the methods of the object's
superclasses.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

My stupid news server won't accept postings with more included than new
text. So I have to add extra text like this. My stupid news server won't
accept postings with more included than new text. So I have to add extra
text like this.
My stupid news server won't accept postings with more included than new
text. So I have to add extra text like this.
My stupid news server won't accept postings with more included than new
text. So I have to add extra text like this.
My stupid news server won't accept postings with more included than new
text. So I have to add extra text like this.

Subject: Re: Object Method validity
Posted by btt on Thu, 14 Aug 2003 11:41:04 GMT

View Forum Message <> Reply to Message

Robert Moss wrote:
> Is there a way to verify the existance of a particular object method?
> Something like
>
> valid = Method_Valid(theObject, "HopedForMethod")
>
> I guess. I've spent a little while rtfm-ing and googling to no avail.

Hi,

Jim Pendelton, IDL wizard, has two handy routines (class_hasmethod and
object_hasmethod) on the RSI user contribution site. These don't tell you if

the method is valid, just if it exists.

http://www.rsinc.com/codebank/search.asp?search=category&amp ;product=IDL&catid=31

Ben

Subject: Re: Object Method validity
Posted by Robert Moss on Fri, 15 Aug 2003 13:54:27 GMT
View Forum Message <> Reply to Message

Ben Tupper wrote:
> Robert Moss wrote:
>
>> Is there a way to verify the existance of a particular object method?
>> Something like
>>
>> valid = Method_Valid(theObject, "HopedForMethod")
>>
>> I guess. I've spent a little while rtfm-ing and googling to no avail.
>
>
>
> Hi,
>
> Jim Pendelton, IDL wizard, has two handy routines (class_hasmethod and
> object_hasmethod) on the RSI user contribution site. These don't tell
> you if the method is valid, just if it exists.
>
> http://www.rsinc.com/codebank/search.asp?search=category&amp ;product=IDL&catid=31
>
>
> Ben
>

Excellent! Thanks, Ben. I'll have to remember to check that site on a
more regular basis.

Robert