Subject: debugging with new variables (dictionary, hash, ...)
Posted by Helder Marchetto on Wed, 14 Jan 2015 22:06:02 GMT
View Forum Message <> Reply to Message

Hi,
I like to debug inserting breakpoints and then I like to use F5 or F6 (step-in and step-over) to go through the code.
If you have some of the new variables in the code, it will be a painful clicking exercise...

Here is a minimal example, that simulates what happens to me quite often...

```
function retValue, inVar1, inVar2
return, {inVar1:inVar1, inVar2:inVar2}
end

pro testDict
d = dictionary('var', 5,'new',6)
print, retValue(d.var, d.new) ;put a breakpoint on this line
end
```

Now run testDict and then suppose you would like to go in the function "retValue" by clicking F5 (step-in). Well, good luck. It's going to take, in this example with two dictionaries, about 93 clicks of F5. Below is the terminal output.

Is there any chance to see all of this disappear in the future?

Regards,
Helder

PS: There are of course ways around this, like putting a new toggle point at the beginning of the procedure/function being called. However, the behavior does not fit with other IDL functions.

```
% Stepped to: DICTIONARY::GETPROPERTY   13
% Stepped to: DICTIONARY::GETPROPERTY   16
% Stepped to: DICTIONARY::GETPROPERTY   18
% Stepped to: DICTIONARY::GETPROPERTY   19
% Stepped to: DICTIONARY::HASKEY   91
% Stepped to: DICTIONARY::HASKEY   93
% Stepped to: DICTIONARY::HASKEY   96
% Stepped to: DICTIONARY::HASKEY   99
% Stepped to: HASH::HASKEY     428
% Stepped to: HASH::HASKEY     430
% Stepped to: HASH::HASKEY     432
% Stepped to: HASH::HASKEY     435
% Stepped to: HASH::HASKEY     437
% Stepped to: HASH::HASKEY     440
% Stepped to: HASH::HASKEY     442
% Stepped to: HASH::HASKEY     443
```

% Stepped to: HASH::HASKEY 444
% Stepped to: HASH::HASKEY 445
% Stepped to: HASH::HASKEY 448
% Stepped to: HASH::HASKEY 449
% Stepped to: HASH::HASKEY 451
% Stepped to: DICTIONARY::GETPROPERTY 20
% Stepped to: DICTIONARY::GET 77
% Stepped to: DICTIONARY::GET 79
% Stepped to: DICTIONARY::GET 82
% Stepped to: DICTIONARY::GET 85
% Stepped to: HASH::GET 213
% Stepped to: HASH::GET 216
% Stepped to: HASH::GET 218
% Stepped to: HASH::GET 221
% Stepped to: HASH::GET 223
% Stepped to: HASH::GET 226
% Stepped to: HASH::GET 227
% Stepped to: HASH::GET 231
% Stepped to: HASH::GET 232
% Stepped to: HASH::GET 233
% Stepped to: HASH::GET 236
% Stepped to: HASH::GET 237
% Stepped to: HASH::GET 238
% Stepped to: HASH::GET 243
% Stepped to: HASH::GET 246
% Stepped to: HASH::GET 247
% Stepped to: HASH::GET 255
% Stepped to: HASH::GET 256
% Stepped to: HASH::GET 261
% Stepped to: HASH::GET 265
% Stepped to: DICTIONARY::GETPROPERTY 37
% Stepped to: DICTIONARY::GETPROPERTY 13
% Stepped to: DICTIONARY::GETPROPERTY 16
% Stepped to: DICTIONARY::GETPROPERTY 18
% Stepped to: DICTIONARY::GETPROPERTY 19
% Stepped to: DICTIONARY::HASKEY 91
% Stepped to: DICTIONARY::HASKEY 93
% Stepped to: DICTIONARY::HASKEY 96
% Stepped to: DICTIONARY::HASKEY 99
% Stepped to: HASH::HASKEY 428
% Stepped to: HASH::HASKEY 430
% Stepped to: HASH::HASKEY 432
% Stepped to: HASH::HASKEY 435
% Stepped to: HASH::HASKEY 437
% Stepped to: HASH::HASKEY 440
% Stepped to: HASH::HASKEY 442
% Stepped to: HASH::HASKEY 443
% Stepped to: HASH::HASKEY 444

```
% Stepped to: HASH::HASKEY      445
% Stepped to: HASH::HASKEY      448
% Stepped to: HASH::HASKEY      449
% Stepped to: HASH::HASKEY      451
% Stepped to: DICTIONARY::GETPROPERTY   20
% Stepped to: DICTIONARY::GET    77
% Stepped to: DICTIONARY::GET    79
% Stepped to: DICTIONARY::GET    82
% Stepped to: DICTIONARY::GET    85
% Stepped to: HASH::GET         213
% Stepped to: HASH::GET         216
% Stepped to: HASH::GET         218
% Stepped to: HASH::GET         221
% Stepped to: HASH::GET         223
% Stepped to: HASH::GET         226
% Stepped to: HASH::GET         227
% Stepped to: HASH::GET         231
% Stepped to: HASH::GET         232
% Stepped to: HASH::GET         233
% Stepped to: HASH::GET         236
% Stepped to: HASH::GET         237
% Stepped to: HASH::GET         238
% Stepped to: HASH::GET         243
% Stepped to: HASH::GET         246
% Stepped to: HASH::GET         247
% Stepped to: HASH::GET         255
% Stepped to: HASH::GET         256
% Stepped to: HASH::GET         261
% Stepped to: HASH::GET         265
% Stepped to: DICTIONARY::GETPROPERTY   37
```

Subject: Re: debugging with new variables (dictionary, hash, ...)
Posted by wlandsman on Wed, 14 Jan 2015 22:19:07 GMT
View Forum Message <> Reply to Message

You could use the .STEPOVER (.SO) command. ("Unlike .STEP, if .STEPOVER executes a statement that calls another routine, the called routine runs until it ends before control returns to interactive mode.")

Or somewhat more tedious you could use .OUT once you enter into HASH or DICTIONARY to get out in one step.

Now you just have to assign .SO to a key (which I assume can be done but don't know how offhand).

--Wayne

On Wednesday, January 14, 2015 at 5:06:04 PM UTC-5, Helder wrote:
> Hi,
> I like to debug inserting breakpoints and then I like to use F5 or F6 (step-in and step-over) to go through the code.
> If you have some of the new variables in the code, it will be a painful clicking exercise...
>
> Here is a minimal example, that simulates what happens to me quite often...
>
> function retValue, inVar1, inVar2
> return, {inVar1:inVar1, inVar2:inVar2}
> end
>
> pro testDict
> d = dictionary('var', 5,'new',6)
> print, retValue(d.var, d.new) ;put a breakpoint on this line
> end
>
> Now run testDict and then suppose you would like to go in the function "retValue" by clicking F5 (step-in). Well, good luck. It's going to take, in this example with two dictionaries, about 93 clicks of F5. Below is the terminal output.
>
> Is there any chance to see all of this disappear in the future?
>
> Regards,
> Helder
>
> PS: There are of course ways around this, like putting a new toggle point at the beginning of the procedure/function being called. However, the behavior does not fit with other IDL functions.
>
> % Stepped to: DICTIONARY::GETPROPERTY 13
> % Stepped to: DICTIONARY::GETPROPERTY 16
> % Stepped to: DICTIONARY::GETPROPERTY 18
> % Stepped to: DICTIONARY::GETPROPERTY 19
> % Stepped to: DICTIONARY::HASKEY 91
> % Stepped to: DICTIONARY::HASKEY 93
> % Stepped to: DICTIONARY::HASKEY 96
> % Stepped to: DICTIONARY::HASKEY 99
> % Stepped to: HASH::HASKEY 428
> % Stepped to: HASH::HASKEY 430
> % Stepped to: HASH::HASKEY 432
> % Stepped to: HASH::HASKEY 435
> % Stepped to: HASH::HASKEY 437
> % Stepped to: HASH::HASKEY 440
> % Stepped to: HASH::HASKEY 442
> % Stepped to: HASH::HASKEY 443
> % Stepped to: HASH::HASKEY 444
> % Stepped to: HASH::HASKEY 445

> % Stepped to: HASH::HASKEY 448
> % Stepped to: HASH::HASKEY 449
> % Stepped to: HASH::HASKEY 451
> % Stepped to: DICTIONARY::GETPROPERTY 20
> % Stepped to: DICTIONARY::GET 77
> % Stepped to: DICTIONARY::GET 79
> % Stepped to: DICTIONARY::GET 82
> % Stepped to: DICTIONARY::GET 85
> % Stepped to: HASH::GET 213
> % Stepped to: HASH::GET 216
> % Stepped to: HASH::GET 218
> % Stepped to: HASH::GET 221
> % Stepped to: HASH::GET 223
> % Stepped to: HASH::GET 226
> % Stepped to: HASH::GET 227
> % Stepped to: HASH::GET 231
> % Stepped to: HASH::GET 232
> % Stepped to: HASH::GET 233
> % Stepped to: HASH::GET 236
> % Stepped to: HASH::GET 237
> % Stepped to: HASH::GET 238
> % Stepped to: HASH::GET 243
> % Stepped to: HASH::GET 246
> % Stepped to: HASH::GET 247
> % Stepped to: HASH::GET 255
> % Stepped to: HASH::GET 256
> % Stepped to: HASH::GET 261
> % Stepped to: HASH::GET 265
> % Stepped to: DICTIONARY::GETPROPERTY 37
> % Stepped to: DICTIONARY::GETPROPERTY 13
> % Stepped to: DICTIONARY::GETPROPERTY 16
> % Stepped to: DICTIONARY::GETPROPERTY 18
> % Stepped to: DICTIONARY::GETPROPERTY 19
> % Stepped to: DICTIONARY::HASKEY 91
> % Stepped to: DICTIONARY::HASKEY 93
> % Stepped to: DICTIONARY::HASKEY 96
> % Stepped to: DICTIONARY::HASKEY 99
> % Stepped to: HASH::HASKEY 428
> % Stepped to: HASH::HASKEY 430
> % Stepped to: HASH::HASKEY 432
> % Stepped to: HASH::HASKEY 435
> % Stepped to: HASH::HASKEY 437
> % Stepped to: HASH::HASKEY 440
> % Stepped to: HASH::HASKEY 442
> % Stepped to: HASH::HASKEY 443
> % Stepped to: HASH::HASKEY 444
> % Stepped to: HASH::HASKEY 445
> % Stepped to: HASH::HASKEY 448

> % Stepped to: HASH::HASKEY 449
> % Stepped to: HASH::HASKEY 451
> % Stepped to: DICTIONARY::GETPROPERTY 20
> % Stepped to: DICTIONARY::GET 77
> % Stepped to: DICTIONARY::GET 79
> % Stepped to: DICTIONARY::GET 82
> % Stepped to: DICTIONARY::GET 85
> % Stepped to: HASH::GET 213
> % Stepped to: HASH::GET 216
> % Stepped to: HASH::GET 218
> % Stepped to: HASH::GET 221
> % Stepped to: HASH::GET 223
> % Stepped to: HASH::GET 226
> % Stepped to: HASH::GET 227
> % Stepped to: HASH::GET 231
> % Stepped to: HASH::GET 232
> % Stepped to: HASH::GET 233
> % Stepped to: HASH::GET 236
> % Stepped to: HASH::GET 237
> % Stepped to: HASH::GET 238
> % Stepped to: HASH::GET 243
> % Stepped to: HASH::GET 246
> % Stepped to: HASH::GET 247
> % Stepped to: HASH::GET 255
> % Stepped to: HASH::GET 256
> % Stepped to: HASH::GET 261
> % Stepped to: HASH::GET 265
> % Stepped to: DICTIONARY::GETPROPERTY 37

Subject: Re: debugging with new variables (dictionary, hash, ...)
Posted by Jim Pendleton on Wed, 14 Jan 2015 22:25:15 GMT
View Forum Message <> Reply to Message

On Wednesday, January 14, 2015 at 3:19:10 PM UTC-7, wlandsman wrote:
> You could use the .STEPOVER (.SO) command. ("Unlike .STEP, if .STEPOVER executes a statement that calls another routine, the called routine runs until it ends before control returns to interactive mode.")
>
> Or somewhat more tedious you could use .OUT once you enter into HASH or DICTIONARY to get out in one step.
>
> Now you just have to assign .SO to a key (which I assume can be done but don't know how offhand).
>
> --Wayne
>
>

> On Wednesday, January 14, 2015 at 5:06:04 PM UTC-5, Helder wrote:
>> Hi,
>> I like to debug inserting breakpoints and then I like to use F5 or F6 (step-in and step-over) to go through the code.
>> If you have some of the new variables in the code, it will be a painful clicking exercise...
>>
>> Here is a minimal example, that simulates what happens to me quite often...
>>
>> function retValue, inVar1, inVar2
>> return, {inVar1:inVar1, inVar2:inVar2}
>> end
>>
>> pro testDict
>> d = dictionary('var', 5,'new',6)
>> print, retValue(d.var, d.new) ;put a breakpoint on this line
>> end
>>
>> Now run testDict and then suppose you would like to go in the function "retValue" by clicking F5 (step-in). Well, good luck. It's going to take, in this example with two dictionaries, about 93 clicks of F5. Below is the terminal output.
>>
>> Is there any chance to see all of this disappear in the future?
>>
>> Regards,
>> Helder
>>
>> PS: There are of course ways around this, like putting a new toggle point at the beginning of the procedure/function being called. However, the behavior does not fit with other IDL functions.
>>
>> % Stepped to: DICTIONARY::GETPROPERTY 13
>> % Stepped to: DICTIONARY::GETPROPERTY 16
>> % Stepped to: DICTIONARY::GETPROPERTY 18
>> % Stepped to: DICTIONARY::GETPROPERTY 19
>> % Stepped to: DICTIONARY::HASKEY 91
>> % Stepped to: DICTIONARY::HASKEY 93
>> % Stepped to: DICTIONARY::HASKEY 96
>> % Stepped to: DICTIONARY::HASKEY 99
>> % Stepped to: HASH::HASKEY 428
>> % Stepped to: HASH::HASKEY 430
>> % Stepped to: HASH::HASKEY 432
>> % Stepped to: HASH::HASKEY 435
>> % Stepped to: HASH::HASKEY 437
>> % Stepped to: HASH::HASKEY 440
>> % Stepped to: HASH::HASKEY 442
>> % Stepped to: HASH::HASKEY 443
>> % Stepped to: HASH::HASKEY 444
>> % Stepped to: HASH::HASKEY 445
>> % Stepped to: HASH::HASKEY 448

>> % Stepped to: HASH::HASKEY 449
>> % Stepped to: HASH::HASKEY 451
>> % Stepped to: DICTIONARY::GETPROPERTY 20
>> % Stepped to: DICTIONARY::GET 77
>> % Stepped to: DICTIONARY::GET 79
>> % Stepped to: DICTIONARY::GET 82
>> % Stepped to: DICTIONARY::GET 85
>> % Stepped to: HASH::GET 213
>> % Stepped to: HASH::GET 216
>> % Stepped to: HASH::GET 218
>> % Stepped to: HASH::GET 221
>> % Stepped to: HASH::GET 223
>> % Stepped to: HASH::GET 226
>> % Stepped to: HASH::GET 227
>> % Stepped to: HASH::GET 231
>> % Stepped to: HASH::GET 232
>> % Stepped to: HASH::GET 233
>> % Stepped to: HASH::GET 236
>> % Stepped to: HASH::GET 237
>> % Stepped to: HASH::GET 238
>> % Stepped to: HASH::GET 243
>> % Stepped to: HASH::GET 246
>> % Stepped to: HASH::GET 247
>> % Stepped to: HASH::GET 255
>> % Stepped to: HASH::GET 256
>> % Stepped to: HASH::GET 261
>> % Stepped to: HASH::GET 265
>> % Stepped to: DICTIONARY::GETPROPERTY 37
>> % Stepped to: DICTIONARY::GETPROPERTY 13
>> % Stepped to: DICTIONARY::GETPROPERTY 16
>> % Stepped to: DICTIONARY::GETPROPERTY 18
>> % Stepped to: DICTIONARY::GETPROPERTY 19
>> % Stepped to: DICTIONARY::HASKEY 91
>> % Stepped to: DICTIONARY::HASKEY 93
>> % Stepped to: DICTIONARY::HASKEY 96
>> % Stepped to: DICTIONARY::HASKEY 99
>> % Stepped to: HASH::HASKEY 428
>> % Stepped to: HASH::HASKEY 430
>> % Stepped to: HASH::HASKEY 432
>> % Stepped to: HASH::HASKEY 435
>> % Stepped to: HASH::HASKEY 437
>> % Stepped to: HASH::HASKEY 440
>> % Stepped to: HASH::HASKEY 442
>> % Stepped to: HASH::HASKEY 443
>> % Stepped to: HASH::HASKEY 444
>> % Stepped to: HASH::HASKEY 445
>> % Stepped to: HASH::HASKEY 448
>> % Stepped to: HASH::HASKEY 449

```
>>  % Stepped to: HASH::HASKEY     451
>>  % Stepped to: DICTIONARY::GETPROPERTY   20
>>  % Stepped to: DICTIONARY::GET    77
>>  % Stepped to: DICTIONARY::GET    79
>>  % Stepped to: DICTIONARY::GET    82
>>  % Stepped to: DICTIONARY::GET    85
>>  % Stepped to: HASH::GET        213
>>  % Stepped to: HASH::GET        216
>>  % Stepped to: HASH::GET        218
>>  % Stepped to: HASH::GET        221
>>  % Stepped to: HASH::GET        223
>>  % Stepped to: HASH::GET        226
>>  % Stepped to: HASH::GET        227
>>  % Stepped to: HASH::GET        231
>>  % Stepped to: HASH::GET        232
>>  % Stepped to: HASH::GET        233
>>  % Stepped to: HASH::GET        236
>>  % Stepped to: HASH::GET        237
>>  % Stepped to: HASH::GET        238
>>  % Stepped to: HASH::GET        243
>>  % Stepped to: HASH::GET        246
>>  % Stepped to: HASH::GET        247
>>  % Stepped to: HASH::GET        255
>>  % Stepped to: HASH::GET        256
>>  % Stepped to: HASH::GET        261
>>  % Stepped to: HASH::GET        265
>>  % Stepped to: DICTIONARY::GETPROPERTY   37
```

There's also a ".out" button that was added to the workbench toolbar in 8.1.

It would be nice if these classes could be shoved down to C code or the IDL source routines could be exposed instead of being hidden in a .sav file. If anyone can't think of something to get me for my birthday, this would do the trick.

Jim P.

Subject: Re: debugging with new variables (dictionary, hash, ...)
Posted by Helder Marchetto on Wed, 14 Jan 2015 22:28:48 GMT
View Forum Message <> Reply to Message

Hi,
I think that .SO and .OUT are the same as F6 and F7.
I tried in the IDLDE and it didn't work.
If you type .SO first, then it will (obviously) step over the line "print, retValue(...)". If you use F5 to step in (don't know what the command line option is), then you can't get out with .SO. With .OUT you go too far out!

Thanks anyway,
Helder

On Wednesday, January 14, 2015 at 11:19:10 PM UTC+1, wlandsman wrote:
> You could use the .STEPOVER (.SO) command. ("Unlike .STEP, if .STEPOVER executes a statement that calls another routine, the called routine runs until it ends before control returns to interactive mode.")
>
> Or somewhat more tedious you could use .OUT once you enter into HASH or DICTIONARY to get out in one step.
>
> Now you just have to assign .SO to a key (which I assume can be done but don't know how offhand).
>
> --Wayne
>
>
> On Wednesday, January 14, 2015 at 5:06:04 PM UTC-5, Helder wrote:
>> Hi,
>> I like to debug inserting breakpoints and then I like to use F5 or F6 (step-in and step-over) to go through the code.
>> If you have some of the new variables in the code, it will be a painful clicking exercise...
>>
>> Here is a minimal example, that simulates what happens to me quite often...
>>
>> function retValue, inVar1, inVar2
>> return, {inVar1:inVar1, inVar2:inVar2}
>> end
>>
>> pro testDict
>> d = dictionary('var', 5,'new',6)
>> print, retValue(d.var, d.new) ;put a breakpoint on this line
>> end
>>
>> Now run testDict and then suppose you would like to go in the function "retValue" by clicking F5 (step-in). Well, good luck. It's going to take, in this example with two dictionaries, about 93 clicks of F5. Below is the terminal output.
>>
>> Is there any chance to see all of this disappear in the future?
>>
>> Regards,
>> Helder
>>
>> PS: There are of course ways around this, like putting a new toggle point at the beginning of the procedure/function being called. However, the behavior does not fit with other IDL functions.
>>
>> % Stepped to: DICTIONARY::GETPROPERTY 13

```
>> % Stepped to: DICTIONARY::GETPROPERTY   16
>> % Stepped to: DICTIONARY::GETPROPERTY   18
>> % Stepped to: DICTIONARY::GETPROPERTY   19
>> % Stepped to: DICTIONARY::HASKEY   91
>> % Stepped to: DICTIONARY::HASKEY   93
>> % Stepped to: DICTIONARY::HASKEY   96
>> % Stepped to: DICTIONARY::HASKEY   99
>> % Stepped to: HASH::HASKEY     428
>> % Stepped to: HASH::HASKEY     430
>> % Stepped to: HASH::HASKEY     432
>> % Stepped to: HASH::HASKEY     435
>> % Stepped to: HASH::HASKEY     437
>> % Stepped to: HASH::HASKEY     440
>> % Stepped to: HASH::HASKEY     442
>> % Stepped to: HASH::HASKEY     443
>> % Stepped to: HASH::HASKEY     444
>> % Stepped to: HASH::HASKEY     445
>> % Stepped to: HASH::HASKEY     448
>> % Stepped to: HASH::HASKEY     449
>> % Stepped to: HASH::HASKEY     451
>> % Stepped to: DICTIONARY::GETPROPERTY   20
>> % Stepped to: DICTIONARY::GET    77
>> % Stepped to: DICTIONARY::GET    79
>> % Stepped to: DICTIONARY::GET    82
>> % Stepped to: DICTIONARY::GET    85
>> % Stepped to: HASH::GET       213
>> % Stepped to: HASH::GET       216
>> % Stepped to: HASH::GET       218
>> % Stepped to: HASH::GET       221
>> % Stepped to: HASH::GET       223
>> % Stepped to: HASH::GET       226
>> % Stepped to: HASH::GET       227
>> % Stepped to: HASH::GET       231
>> % Stepped to: HASH::GET       232
>> % Stepped to: HASH::GET       233
>> % Stepped to: HASH::GET       236
>> % Stepped to: HASH::GET       237
>> % Stepped to: HASH::GET       238
>> % Stepped to: HASH::GET       243
>> % Stepped to: HASH::GET       246
>> % Stepped to: HASH::GET       247
>> % Stepped to: HASH::GET       255
>> % Stepped to: HASH::GET       256
>> % Stepped to: HASH::GET       261
>> % Stepped to: HASH::GET       265
>> % Stepped to: DICTIONARY::GETPROPERTY   37
>> % Stepped to: DICTIONARY::GETPROPERTY   13
>> % Stepped to: DICTIONARY::GETPROPERTY   16
```

```
>> % Stepped to: DICTIONARY::GETPROPERTY   18
>> % Stepped to: DICTIONARY::GETPROPERTY   19
>> % Stepped to: DICTIONARY::HASKEY   91
>> % Stepped to: DICTIONARY::HASKEY   93
>> % Stepped to: DICTIONARY::HASKEY   96
>> % Stepped to: DICTIONARY::HASKEY   99
>> % Stepped to: HASH::HASKEY     428
>> % Stepped to: HASH::HASKEY     430
>> % Stepped to: HASH::HASKEY     432
>> % Stepped to: HASH::HASKEY     435
>> % Stepped to: HASH::HASKEY     437
>> % Stepped to: HASH::HASKEY     440
>> % Stepped to: HASH::HASKEY     442
>> % Stepped to: HASH::HASKEY     443
>> % Stepped to: HASH::HASKEY     444
>> % Stepped to: HASH::HASKEY     445
>> % Stepped to: HASH::HASKEY     448
>> % Stepped to: HASH::HASKEY     449
>> % Stepped to: HASH::HASKEY     451
>> % Stepped to: DICTIONARY::GETPROPERTY   20
>> % Stepped to: DICTIONARY::GET   77
>> % Stepped to: DICTIONARY::GET   79
>> % Stepped to: DICTIONARY::GET   82
>> % Stepped to: DICTIONARY::GET   85
>> % Stepped to: HASH::GET       213
>> % Stepped to: HASH::GET       216
>> % Stepped to: HASH::GET       218
>> % Stepped to: HASH::GET       221
>> % Stepped to: HASH::GET       223
>> % Stepped to: HASH::GET       226
>> % Stepped to: HASH::GET       227
>> % Stepped to: HASH::GET       231
>> % Stepped to: HASH::GET       232
>> % Stepped to: HASH::GET       233
>> % Stepped to: HASH::GET       236
>> % Stepped to: HASH::GET       237
>> % Stepped to: HASH::GET       238
>> % Stepped to: HASH::GET       243
>> % Stepped to: HASH::GET       246
>> % Stepped to: HASH::GET       247
>> % Stepped to: HASH::GET       255
>> % Stepped to: HASH::GET       256
>> % Stepped to: HASH::GET       261
>> % Stepped to: HASH::GET       265
>> % Stepped to: DICTIONARY::GETPROPERTY   37
```

Subject: Re: debugging with new variables (dictionary, hash, ...)
Posted by Helder Marchetto on Wed, 14 Jan 2015 22:35:20 GMT
View Forum Message <> Reply to Message

I can only offer a "happy birthday". I have no chance to access the code.
I think that the following are the same (at least in my default workbench configuration):
Keyboard = workbench = command line
F5 = "In" = ???
F6 = "Over" = .SO
F7 = "Out" = .OUT

There is also the "don't try this at home" option.
While in the dictionary or hash procedure, try what happens when you press Ctrl+Shift+F6 (=Skip). You will generate some "unexpected" errors like:

% SCOPE_VARFETCH: String expression required in this context: KEY.

Cheers,
Helder

On Wednesday, January 14, 2015 at 11:25:16 PM UTC+1, Jim P wrote:
> On Wednesday, January 14, 2015 at 3:19:10 PM UTC-7, wlandsman wrote:
>> You could use the .STEPOVER (.SO) command. ("Unlike .STEP, if .STEPOVER executes a statement that calls another routine, the called routine runs until it ends before control returns to interactive mode.")
>>
>> Or somewhat more tedious you could use .OUT once you enter into HASH or DICTIONARY to get out in one step.
>>
>> Now you just have to assign .SO to a key (which I assume can be done but don't know how offhand).
>>
>> --Wayne
>>
>>
>> On Wednesday, January 14, 2015 at 5:06:04 PM UTC-5, Helder wrote:
>>> Hi,
>>> I like to debug inserting breakpoints and then I like to use F5 or F6 (step-in and step-over) to go through the code.
>>> If you have some of the new variables in the code, it will be a painful clicking exercise...
>>>
>>> Here is a minimal example, that simulates what happens to me quite often...
>>>
>>> function retValue, inVar1, inVar2
>>> return, {inVar1:inVar1, inVar2:inVar2}
>>> end
>>>
>>> pro testDict
>>> d = dictionary('var', 5,'new',6)

>>> print, retValue(d.var, d.new) ;put a breakpoint on this line
>>> end
>>>
>>> Now run testDict and then suppose you would like to go in the function "retValue" by clicking F5 (step-in). Well, good luck. It's going to take, in this example with two dictionaries, about 93 clicks of F5. Below is the terminal output.
>>>
>>> Is there any chance to see all of this disappear in the future?
>>>
>>> Regards,
>>> Helder
>>>
>>> PS: There are of course ways around this, like putting a new toggle point at the beginning of the procedure/function being called. However, the behavior does not fit with other IDL functions.
>>>
>>> % Stepped to: DICTIONARY::GETPROPERTY 13
>>> % Stepped to: DICTIONARY::GETPROPERTY 16
>>> % Stepped to: DICTIONARY::GETPROPERTY 18
>>> % Stepped to: DICTIONARY::GETPROPERTY 19
>>> % Stepped to: DICTIONARY::HASKEY 91
>>> % Stepped to: DICTIONARY::HASKEY 93
>>> % Stepped to: DICTIONARY::HASKEY 96
>>> % Stepped to: DICTIONARY::HASKEY 99
>>> % Stepped to: HASH::HASKEY 428
>>> % Stepped to: HASH::HASKEY 430
>>> % Stepped to: HASH::HASKEY 432
>>> % Stepped to: HASH::HASKEY 435
>>> % Stepped to: HASH::HASKEY 437
>>> % Stepped to: HASH::HASKEY 440
>>> % Stepped to: HASH::HASKEY 442
>>> % Stepped to: HASH::HASKEY 443
>>> % Stepped to: HASH::HASKEY 444
>>> % Stepped to: HASH::HASKEY 445
>>> % Stepped to: HASH::HASKEY 448
>>> % Stepped to: HASH::HASKEY 449
>>> % Stepped to: HASH::HASKEY 451
>>> % Stepped to: DICTIONARY::GETPROPERTY 20
>>> % Stepped to: DICTIONARY::GET 77
>>> % Stepped to: DICTIONARY::GET 79
>>> % Stepped to: DICTIONARY::GET 82
>>> % Stepped to: DICTIONARY::GET 85
>>> % Stepped to: HASH::GET 213
>>> % Stepped to: HASH::GET 216
>>> % Stepped to: HASH::GET 218
>>> % Stepped to: HASH::GET 221
>>> % Stepped to: HASH::GET 223
>>> % Stepped to: HASH::GET 226
>>> % Stepped to: HASH::GET 227

```
>>> % Stepped to: HASH::GET       231
>>> % Stepped to: HASH::GET       232
>>> % Stepped to: HASH::GET       233
>>> % Stepped to: HASH::GET       236
>>> % Stepped to: HASH::GET       237
>>> % Stepped to: HASH::GET       238
>>> % Stepped to: HASH::GET       243
>>> % Stepped to: HASH::GET       246
>>> % Stepped to: HASH::GET       247
>>> % Stepped to: HASH::GET       255
>>> % Stepped to: HASH::GET       256
>>> % Stepped to: HASH::GET       261
>>> % Stepped to: HASH::GET       265
>>> % Stepped to: DICTIONARY::GETPROPERTY   37
>>> % Stepped to: DICTIONARY::GETPROPERTY   13
>>> % Stepped to: DICTIONARY::GETPROPERTY   16
>>> % Stepped to: DICTIONARY::GETPROPERTY   18
>>> % Stepped to: DICTIONARY::GETPROPERTY   19
>>> % Stepped to: DICTIONARY::HASKEY   91
>>> % Stepped to: DICTIONARY::HASKEY   93
>>> % Stepped to: DICTIONARY::HASKEY   96
>>> % Stepped to: DICTIONARY::HASKEY   99
>>> % Stepped to: HASH::HASKEY     428
>>> % Stepped to: HASH::HASKEY     430
>>> % Stepped to: HASH::HASKEY     432
>>> % Stepped to: HASH::HASKEY     435
>>> % Stepped to: HASH::HASKEY     437
>>> % Stepped to: HASH::HASKEY     440
>>> % Stepped to: HASH::HASKEY     442
>>> % Stepped to: HASH::HASKEY     443
>>> % Stepped to: HASH::HASKEY     444
>>> % Stepped to: HASH::HASKEY     445
>>> % Stepped to: HASH::HASKEY     448
>>> % Stepped to: HASH::HASKEY     449
>>> % Stepped to: HASH::HASKEY     451
>>> % Stepped to: DICTIONARY::GETPROPERTY   20
>>> % Stepped to: DICTIONARY::GET    77
>>> % Stepped to: DICTIONARY::GET    79
>>> % Stepped to: DICTIONARY::GET    82
>>> % Stepped to: DICTIONARY::GET    85
>>> % Stepped to: HASH::GET       213
>>> % Stepped to: HASH::GET       216
>>> % Stepped to: HASH::GET       218
>>> % Stepped to: HASH::GET       221
>>> % Stepped to: HASH::GET       223
>>> % Stepped to: HASH::GET       226
>>> % Stepped to: HASH::GET       227
>>> % Stepped to: HASH::GET       231
```

>>> % Stepped to: HASH::GET 232
>>> % Stepped to: HASH::GET 233
>>> % Stepped to: HASH::GET 236
>>> % Stepped to: HASH::GET 237
>>> % Stepped to: HASH::GET 238
>>> % Stepped to: HASH::GET 243
>>> % Stepped to: HASH::GET 246
>>> % Stepped to: HASH::GET 247
>>> % Stepped to: HASH::GET 255
>>> % Stepped to: HASH::GET 256
>>> % Stepped to: HASH::GET 261
>>> % Stepped to: HASH::GET 265
>>> % Stepped to: DICTIONARY::GETPROPERTY 37
>
> There's also a ".out" button that was added to the workbench toolbar in 8.1.
>
> It would be nice if these classes could be shoved down to C code or the IDL source routines could be exposed instead of being hidden in a .sav file. If anyone can't think of something to get me for my birthday, this would do the trick.
>
> Jim P.

Subject: Re: debugging with new variables (dictionary, hash, ...)
Posted by Fabzi on Thu, 15 Jan 2015 12:06:45 GMT
View Forum Message <> Reply to Message

Hi Helder,

On 14.01.2015 23:06, Helder wrote:
> it will be a painful clicking exercise...

yes, I reported the same problem here long ago:
 https://groups.google.com/forum/#!topic/comp.lang.idl-pvwave /9ATWbQ7bs2o

I agree that this is very annoying in practice, but maybe it's just us...

Fabien

Subject: Re: debugging with new variables (dictionary, hash, ...)
Posted by Helder Marchetto on Thu, 15 Jan 2015 12:27:17 GMT
View Forum Message <> Reply to Message

Hi Fabien,
sorry, but I didn't google your post. It's exactly the same problem I have and I find it quite annoying.

Specially because I have some pro code in some file somewhere in my directory/project and when I want to debug such code, it's easier to put a breakpoint on such lines and then "step-in". However, I have to deal with loads of unnecessary clicking...

A workaround would be great.

At the time Jim wanted to convert to c and Chris thought this could be solved with the right button (F6) or that there is no need to step-into build-in IDL functions like mean(). I support Jim's idea, but it's not up to me :-)
As for Chris comments, unfortunately these don't work and most of the time it is our procedures we want to debug, not IDLs (by good luck).

So, is there any update on this from Exelis?

Regards,
Helder

On Thursday, January 15, 2015 at 1:06:49 PM UTC+1, Fabien wrote:
> Hi Helder,
>
> On 14.01.2015 23:06, Helder wrote:
>> it will be a painful clicking exercise...
>
> yes, I reported the same problem here long ago:
> https://groups.google.com/forum/#!topic/comp.lang.idl-pvwave /9ATWbQ7bs2o
>
> I agree that this is very annoying in practice, but maybe it's just us...
>
> Fabien

Subject: Re: debugging with new variables (dictionary, hash, ...)
Posted by wlandsman on Thu, 15 Jan 2015 13:27:52 GMT
View Forum Message <> Reply to Message

On Wednesday, January 14, 2015 at 5:28:50 PM UTC-5, Helder wrote:

> If you type .SO first, then it will (obviously) step over the line "print, retValue(...)". If you use F5 to step in (don't know what the command line option is), then you can't get out with .SO. With .OUT you go too far out!
>
> Thanks anyway,
> Helder

I don't understand this comment. .SO will *not* step over the line "print, retValue(...)". Stepover is perhaps an unfortunate name. It does not skip or "step over" any lines. It takes

one step and *if* you enter another procedure in that step it continues until you return (e.g exit the HASH call). This appears to be exactly what you are asking for.

As for .OUT you would use it once you step into the HASH routine to get out of the HASH routine in one keystroke. Of course, by using .SO you can avoid ever entering the hash routine in the first place.

Subject: Re: debugging with new variables (dictionary, hash, ...)
Posted by Helder Marchetto on Thu, 15 Jan 2015 14:07:55 GMT
View Forum Message <> Reply to Message

Hi,
I just found that the following combination of F-keys works for my purpose:
F5 and then F7. This can be done from the command line as .step and .out.
Wayne: thanks for the tip.

I'm fine with this (two key strokes are for sure better than 90 and just a bit worse than 1), however, for consistency, it would be nice if these functions (hash, dictionary,...) would be implemented in the compiler in such a way that one cannot go into them (.step).

Thanks and sorry for bothering with what seems like a trivial issue, but has driven me mad (and eroded my F5 key down of ~1mm).

Cheers,
Helder

On Thursday, January 15, 2015 at 2:27:54 PM UTC+1, wlandsman wrote:
> On Wednesday, January 14, 2015 at 5:28:50 PM UTC-5, Helder wrote:
>
>> If you type .SO first, then it will (obviously) step over the line "print, retValue(...)". If you use F5 to step in (don't know what the command line option is), then you can't get out with .SO. With .OUT you go too far out!
>>
>> Thanks anyway,
>> Helder
>
> I don't understand this comment. .SO will *not* step over the line "print, retValue(...)". Stepover is perhaps an unfortunate name. It does not skip or "step over" any lines. It takes one step and *if* you enter another procedure in that step it continues until you return (e.g exit the HASH call). This appears to be exactly what you are asking for.
>
> As for .OUT you would use it once you step into the HASH routine to get out of the HASH routine in one keystroke. Of course, by using .SO you can avoid ever entering the hash routine in the first place.

Subject: Re: debugging with new variables (dictionary, hash, ...)
Posted by Jim Pendleton on Thu, 15 Jan 2015 15:45:47 GMT
View Forum Message <> Reply to Message

On Thursday, January 15, 2015 at 7:07:57 AM UTC-7, Helder wrote:
> Hi,
> I just found that the following combination of F-keys works for my purpose:
> F5 and then F7. This can be done from the command line as .step and .out.
> Wayne: thanks for the tip.
>
> I'm fine with this (two key strokes are for sure better than 90 and just a bit worse than 1), however, for consistency, it would be nice if these functions (hash, dictionary,...) would be implemented in the compiler in such a way that one cannot go into them (.step).
>
> Thanks and sorry for bothering with what seems like a trivial issue, but has driven me mad (and eroded my F5 key down of ~1mm).
>
> Cheers,
> Helder
>
>
> On Thursday, January 15, 2015 at 2:27:54 PM UTC+1, wlandsman wrote:
>> On Wednesday, January 14, 2015 at 5:28:50 PM UTC-5, Helder wrote:
>>
>>> If you type .SO first, then it will (obviously) step over the line "print, retValue(...)". If you use F5 to step in (don't know what the command line option is), then you can't get out with .SO. With .OUT you go too far out!
>>>
>>> Thanks anyway,
>>> Helder
>>
>> I don't understand this comment. .SO will *not* step over the line "print, retValue(...)". Stepover is perhaps an unfortunate name. It does not skip or "step over" any lines. It takes one step and *if* you enter another procedure in that step it continues until you return (e.g exit the HASH call). This appears to be exactly what you are asking for.
>>
>> As for .OUT you would use it once you step into the HASH routine to get out of the HASH routine in one keystroke. Of course, by using .SO you can avoid ever entering the hash routine in the first place.

On the plus side, notice that as of 8.4 you can now see line numbers from .sav files echoed to the console as you step through the code. This is not particularly helpful in this example, but if you were the keeper of the original .pro code for your own runtime application that you have distributed to others this is a positive development.