On Wednesday, February 18, 2015 at 2:47:30 PM UTC-7, JDS wrote:
>>
>> The crash is fixed in IDL 8.4.1. Thanks for reporting it!
>> -Chris
>
> Thanks.
>
>> p.s. I can't quite wrap my head around how the auto-instantiation would work. More examples?
>
> If you are constructing some nested HASH structure, now you must say:
>
> a=hash()
> a['key']=hash()
> a['key','sub1']=hash()
> a['key','sub1','sub2']=hash()
> a['key','sub1','sub2','value']=1.0
>
> This might more typically occur inside a loop pulling keys and values from various locations. This is also by the way why I was using a=hash() & c=(a['b']=hash()) -- to make this sort of construction slightly less painful.
>
> Auto-instantiation means that any hash key which references an undefined value *on assignment* will cause that value to be initialized as a HASH object instead of just saying "key does not exist" and aborting. If that were in place, the above would simply be:
>
> a=hash()
> a['key','sub1','sub2','value']=1.0
>
> This comes up quite a bit when attempting to populate deeply nested HASH structures. You end up with code sprinkled with lots of useless tests like:
>
> if ~a[key1].hasKey(key2) then a[key1,key2]=hash()
>
> With auto-instantiation, these statements would be implicit. It also would make Perl programmers happy ;).
>
> Thanks again,
>
> JD

Okay, I like that. I was able to hack it in with 3 lines of code.

IDL> a = hash()

```
IDL> a['a1','b1','c1','d1','e1','f1','g1','h1'] = 5
IDL> a
{
    "a1": {
       "b1": {
          "c1": {
             "d1": {
                "e1": {
                   "f1": {
                      "g1": {
                         "h1": 5
                      }
                   }
                }
             }
          }
       }
    }
}
```

Hmmm. Not sure how I can get the code to you...

-Chris