Subject: IDL8.4 hard crash
Posted by JDS on Wed, 18 Feb 2015 20:37:44 GMT
View Forum Message <> Reply to Message

Try:

IDL> a=hash() & c=(a['b']=hash())

I believe auto-instantiation would make HASH/DICT/etc. much more useful. I.e. I ought to be able to say:

IDL> a=hash() & a['b','c']='test'

and have a['b'] auto-initialized as an empty hash.

JD

Subject: Re: IDL8.4 hard crash
Posted by bill.dman on Wed, 18 Feb 2015 20:55:38 GMT
View Forum Message <> Reply to Message

On Wednesday, February 18, 2015 at 3:37:46 PM UTC-5, JDS wrote:
> Try:
>
> IDL> a=hash() & c=(a['b']=hash())
>
> I believe auto-instantiation would make HASH/DICT/etc. much more useful. I.e. I ought to be able to say:
>
> IDL> a=hash() & a['b','c']='test'
>
> and have a['b'] auto-initialized as an empty hash.
>
> JD

Crashes on OSx 10.9.5, on RHEL 6.6 gives:
% Key must be a scalar string or number.
% Execution halted at: $MAIN$

Subject: Re: IDL8.4 hard crash
Posted by chris_torrence@NOSPAM on Wed, 18 Feb 2015 21:18:05 GMT
View Forum Message <> Reply to Message

On Wednesday, February 18, 2015 at 1:55:39 PM UTC-7, bill...@gmail.com wrote:
> On Wednesday, February 18, 2015 at 3:37:46 PM UTC-5, JDS wrote:

>> Try:
>>
>> IDL> a=hash() & c=(a['b']=hash())
>>
>> I believe auto-instantiation would make HASH/DICT/etc. much more useful. I.e. I ought to be able to say:
>>
>> IDL> a=hash() & a['b','c']='test'
>>
>> and have a['b'] auto-initialized as an empty hash.
>>
>> JD
>
> Crashes on OSx 10.9.5, on RHEL 6.6 gives:
> % Key must be a scalar string or number.
> % Execution halted at: $MAIN$

The crash is fixed in IDL 8.4.1. Thanks for reporting it!
-Chris
p.s. I can't quite wrap my head around how the auto-instantiation would work. More examples?

Subject: Re: IDL8.4 hard crash
Posted by JDS on Wed, 18 Feb 2015 21:47:28 GMT

View Forum Message <> Reply to Message

>
> 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

Subject: Re: IDL8.4 hard crash
Posted by chris_torrence@NOSPAM on Wed, 18 Feb 2015 22:18:22 GMT

View Forum Message <> Reply to Message

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

Subject: Re: IDL8.4 hard crash
Posted by Fabzi on Wed, 18 Feb 2015 22:21:12 GMT
View Forum Message <> Reply to Message

On 18.02.2015 22:47, JDS wrote:

> It also would make Perl programmers happy;).

In python this is called "defaultdict":

https://docs.python.org/2/library/collections.html#collectio ns.defaultdict

Cheers,

Fabien

Subject: Re: IDL8.4 hard crash
Posted by Fabzi on Thu, 19 Feb 2015 08:22:56 GMT
View Forum Message <> Reply to Message

On 18.02.2015 23:18, Chris Torrence wrote:

```
>  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
>                              }
>                          }
>                      }
>                  }
>              }
>          }
>      }
>  }
```

That's cool! But it doesn't have to be a hash() as default. You could also need something like:

```
a = hash(DEFAULT=list())
a['key']->add, 1
```

Fabien

Subject: Re: IDL8.4 hard crash
Posted by chris_torrence@NOSPAM on Thu, 19 Feb 2015 14:40:31 GMT
View Forum Message <> Reply to Message

On Thursday, February 19, 2015 at 1:23:01 AM UTC-7, Fabien wrote:

```
>  On 18.02.2015 23:18, Chris Torrence wrote:
>>  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
>>                              }
>>                          }
>>                      }
>>                  }
>>              }
>>          }
>>      }
>>  }
>
>  That's cool! But it doesn't have to be a hash() as default. You could
>  also need something like:
>
>  a = hash(DEFAULT=list())
>  a['key']->add, 1
>
>  Fabien
```

But I think in this case, since we're indexing using strings, then we know that we want a hash for the sub-container. I'd hate to complicate it further with a keyword that I have to document.
-Chris

Subject: Re: IDL8.4 hard crash
Posted by Fabzi on Thu, 19 Feb 2015 15:19:38 GMT
View Forum Message <> Reply to Message

On 19.02.2015 15:40, Chris Torrence wrote:

```
>  But I think in this case, since we're indexing using strings,
```

>then we know that we want a hash for the sub-container. I'd hate
> to complicate it further with a keyword that I have to document.

The argument of the string indexes is true if you have more than one
nested level.

Just in case you are thinking of a Hash() improvement for a future IDL,
it would be good not to be limited to default hashes only, as does
python's defaultdict. The roblem of course is that my example is flawed:

a = hash(DEFAULT=list())

is not okay. I shouldn't give an instance of list but rather a "type"
list or so:

a = hash(DEFAULT='list')

Subject: Hash auto-instantiation (was IDL8.4 hard crash)
Posted by chris_torrence@NOSPAM on Thu, 19 Feb 2015 18:28:39 GMT
View Forum Message <> Reply to Message

On Thursday, February 19, 2015 at 8:19:42 AM UTC-7, Fabien wrote:
> On 19.02.2015 15:40, Chris Torrence wrote:
>> But I think in this case, since we're indexing using strings,
>> then we know that we want a hash for the sub-container. I'd hate
>> to complicate it further with a keyword that I have to document.
>
> The argument of the string indexes is true if you have more than one
> nested level.
>
> Just in case you are thinking of a Hash() improvement for a future IDL,
> it would be good not to be limited to default hashes only, as does
> python's defaultdict. The roblem of course is that my example is flawed:
>
> a = hash(DEFAULT=list())
>
> is not okay. I shouldn't give an instance of list but rather a "type"
> list or so:
>
> a = hash(DEFAULT='list')

Actually, in my code I am creating the new container based upon my own class. So if you have an IDL Dictionary it will create a Dictionary for the sub-containers. Similarly for the OrderedHash class.

For example:
d = Dictionary()

```
d['a','b','c'] = 5
help, d['a']
<Expression>    DICTIONARY  <ID=6216  NELEMENTS=1>
```

Speaking of dictionaries, now I'm wondering whether it should work for the "dot" notation as well as the brackets:

```
d = Dictionary()
d.a.b.c = 5    ; should this auto-instantiate???
```

I'm thinking that it probably should do the same thing...

-Chris