Subject: Re: Copying a hash
Posted by Paul Van Delst[1] on Mon, 06 Aug 2012 21:44:37 GMT
View Forum Message <> Reply to Message

On 08/06/12 16:54, Matt wrote:
> Hi All,
>
> Does anyone know if there's a simple way that I can make a copy of a hash, which I can then edit independently of the
> original? For example, it seems that, like a pointer, changes that I make to the copy are also applied to the
> original:
>
> IDL> original=hash('A', [1, 2]) IDL> copy=original IDL> copy['A', 1]=10 IDL> print, copy A: 1 10 IDL>
> print, original A: 1 10
>
> I can copy to a new hash key-by-key:
>
> copy=hash() foreach variable, original, key do copy[key]=original[key]
>
> Which works fine, unless one of the elements in the hash is itself a hash, then I end up with the same problem one
> level down.
>
> Is there something simple I'm missing here?

This is what the documentation says:

-----%<-----
To create a new hash variable whose elements are copies of the values in the original hash, you could use the following:

```
  newHash = HASH(origHash.Keys(), origHash.Values())
```

Another method to copy a hash is to use array syntax to copy all of the elements:

```
  newHash = origHash[*]
```

This is equivalent to using HASH(origHash.Keys(), origHash.Values()) and is provided as a programming shortcut.

For example:

```
  hash1 = HASH('key1', 1, 'key2', 2)
  hash2 = hash1[*]
  hash2['key1'] = 'hello'
  HELP, hash1['key1'], hash2['key2']
```

IDL Prints:

```
  <Expression> STRING = 1
  <Expression> STRING = 'hello'
```

Note that the value in hash1 remains unchanged.
-----%<-----

There's no mention of what happens if a hash value is itself a hash though.

cheers,

paulv