Subject: Any interest in an IDL to Python interface?
Posted by Jason Ferrara on Thu, 28 Feb 2008 16:00:30 GMT
View Forum Message <> Reply to Message

We're thinking of coming out with product that acts as a bridge between IDL and Python, and are trying to get an idea of how much of a demand there is for this sort of thing.

It would make Python modules usable directly from IDL.

Some simple usage examples, meant to show how the interface works, rather than why you might want to use Python from IDL.

Using the Python Imaging Library to load an image, rotate it, and then place it in an IDL array.

```
IDL> numpy=pyimport('numpy')
Loaded DLM: PYTHONFROMIDL.
IDL> pilimage=pyimport('PIL.Image')
IDL> img=pilimage->open('scan.jpeg')
IDL> img=img->rotate(30)
IDL> imgarr=numpy->array(img)
IDL> help, imgarr
IMGARR          BYTE      = Array[3, 850, 864]
IDL> tv, imgarr, /true, order=1
```

Defining and calling an arbitrary python function.

```
IDL> py=pyimport('__main__')
IDL> py->exec, "def mulbytwo(a):"+string(10b)+"    return [x*2 for x
in a]"
IDL> print, py->mulbytwo([1,2,3,4,5])
          2          4          6          8         10
```

Features of the interface:

Python objects (including modules) appear in IDL as IDL objects.

Automatic conversion of method parameters from IDL variables to the appropriate python type.

Automatic conversion of return values to IDL types.

The Python environment runs in the same process as IDL, so parameter passing is fast.

Automatic garbage collection of IDL objects that represent Python objects, so calling OBJ_DESTROY is not required. This makes the objects behave more Python like, so that you can do things like

"img=((pilimage->open('scan.jpeg'))->rotate(30))->convert('L') "
without leaking objects or having to call HEAP_GC.

Would anyone find this useful?

Thanks

Jason Ferrara
Jacquette Consulting, Inc.
