Managing BIG-IP Routes with the Python Bigsuds Library

Jason Rahm, 2013-05-12

Perusing the Q&A section of DevCentral a couple days ago, I noticed a question on adding routes from community member Richard:

> Is there a smart way to add a bunch of routes that doesn't mandate manual entry... (The JRahm paraphrase)

There were a couple great answers on using merge or copy/paste, but this jogged the memory that "hey, I've done that before!" A few yeas ago I was just learning python and how to use the iControl API and wrote a three part series on this very thing:

1. pyControl Apps #1 - BIG-IP Routing Table
2. pyControl Apps #2 - Deleting BIG-IP Routes
3. pyControl Apps #3 - Adding BIG-IP Routes

That works, but pycontrol v1 (these scripts) and even pycontrol v2 are no longer supported/developed since the bigsuds library was introduced late last year. So I decided to update the script to support the bigsuds library, the python argparse module over the originally used and discontinued optparse module, and finally, using the RouteTableV2 iControl interface instead of the deprecated RouteTable interface.

The Main Loop

First, the code:

```
if __name__ == "__main__":

        import bigsuds as pc
        import getpass
        import argparse

        parser = argparse.ArgumentParser()

        parser.add_argument("-s",  "--system", required=True)
        parser.add_argument("-u", "--username", required=True)
        parser.add_argument("-d",  "--delRoute")
        parser.add_argument("-a",  "--addRoute")
        args = vars(parser.parse_args())

        print "%s, enter your " % args['username'],
        upass = getpass.getpass()

        b = pc.BIGIP(args['system'], args['username'], upass)

        if args['delRoute']:
            del_tmmRoutes(b.Networking.RouteTable,  args['delRoute'])

        if args['addRoute']:
            add_tmmRoutes(b.Networking.RouteTableV2, args['addRoute'], b.LocalLB.Pool, b.Networking.V

        get_tmmRoutes(b.Networking.RouteTableV2)
```

The main loop remains unchanged in structure. First, I update the iControl reference for the bigsuds library instead of the pycontrol one and update the instantiation arguments. Next, I needed to swap out optparse in favor of argparse to handle the command line arguments for the script. Finally, I updated the iControl references from RouteTable to RouteTableV2, except in the reference to the delete route function. More on that below.

The Delete Route Function

Beginning in TMOS version 11, the RouteTable iControl interface was deprecated and RouteTableV2 introduced. However, the parameters for the delete_static_route method changed from an address/mask structure to a string, expecting the route name instead of the route address/mask. This is fine if you know the route names, but I haven't yet found an easy way to remove routes solely based on the routing information in the new interface.

```
def del_tmmRoutes(obj,  filename):

    routefile = open(filename,  'r')
    headers = routefile.readline().strip().split(',')
    stored_rows = []

    for line in routefile:
        route = line.strip().split(',')
        stored_rows.append(dict(zip(headers,  route)))


    for route in stored_rows:
        obj.delete_static_route(routes = [route])
        print "Deleting Route %s/%s" % (route['destination'], route['netmask'])
```

The Add Route Function

This function just needed a little updating for the change from the add_static_route method to the create_static_route method. Now that routes require names, I had to account for the slice of headers/data I took from the routes in the text file. Structurally there were no changes. One thing I don't yet have working in the new interface is the reject route, so I've removed that code from the function displayed below, though the get routes function below will still display any reject routes in the system.

```
def add_tmmRoutes(obj,  filename, pool, vlan):
    pools = pool.get_list()
    vlans = vlan.get_list()
    routefile = open(filename, 'r')
    headers = routefile.readline().strip().split(',')
    rt_hdrs = headers[:1]
    dest_hdrs = headers[1:3]
    atr_hdrs = ['gateway','pool_name','vlan_name']
    rts = []
    dests = []
    atrs = []
    for line in routefile:
        ln = line.strip().split(',')
        rt_name = ln[:1]
        dest = ln[1:3]
        atr = ln[-2:]
        if atr[0] == 'pool':
            if atr[1] in pools:
                atrs.append(dict(zip(atr_hdrs, ['',atr[1],''])))
```

```
                dests.append(dict(zip(dest_hdrs, dest)))
                rts.append(rt_name)
            else:
                print "Pool ", atr[1], " does not exist"
        elif atr[0] == 'vlan':
            if atr[1] in vlans:
                atrs.append(dict(zip(atr_hdrs, ['','',atr[1]])))
                dests.append(dict(zip(dest_hdrs, dest)))
                rts.append(rt_name)
            else:
                print "Vlan ", atr[1], " does not exist"
        elif atr[0] == 'gateway':
            atrs.append(dict(zip(atr_hdrs, [atr[1],'',''])))
            dests.append(dict(zip(dest_hdrs, dest)))
            rts.append(rt_name)

    combined = zip(rts, dests, atrs)

    for x in combined:
        xl = list(x)
        obj.create_static_route(routes = xl[0], destinations = [xl[1]], attributes = [xl[2]])
```

The IP Sorting Function

This function isn't necessary and can be removed if desired, I just included it to sort all the routes properly. The only update here was to change the data reference (i[2]['address']) in the list comprehension.

```
def sort_ip_dict(ip_list):
    from IPy import IP
    ipl = [ (IP(i[2]['address']).int(),  i) for i in ip_list]
    ipl.sort()
    return [ip[1] for ip in ipl]
```

The Get Routes Function

In this function I had to update a couple of the methods specific to the RouteTableV2 interface, work with some changing data types between the ZSI (pycontrol) and suds (bigsuds) libraries. I updated the output a little as well.

```
def get_tmmRoutes(obj):

    try:
        tmmStatic = obj.get_static_route_list()
        tmmRtType = obj.get_static_route_type(routes = tmmStatic)
        tmmRtDest = obj.get_static_route_destination(routes = tmmStatic)

    except:
        "Unable to fetch route information - check trace log"

    combined = zip(tmmStatic,  tmmRtType, tmmRtDest)
    combined = [list(a) for a in combined]

    ldict_gw_ip = []
    ldict_gw_pool = []
    ldict_gw_vlan = []
    ldict_gw_reject = []
```

```
    for x in combined:
            if x[1] == 'ROUTE_TYPE_GATEWAY':
                x.append(obj.get_static_route_gateway(routes = [x[0]])[0])
                ldict_gw_ip.append(x)
            if x[1] == 'ROUTE_TYPE_POOL':
                x.append(obj.get_static_route_pool(routes = [x[0]])[0])
                ldict_gw_pool.append(x)
            if x[1] == 'ROUTE_TYPE_INTERFACE':
                x.append(obj.get_static_route_vlan(routes = [x[0]])[0])
                ldict_gw_vlan.append(x)
            if x[1] == 'ROUTE_TYPE_REJECT':
                ldict_gw_reject.append(x)

    gw_ip = sort_ip_dict(ldict_gw_ip)
    gw_pool = sort_ip_dict(ldict_gw_pool)
    gw_vlan = sort_ip_dict(ldict_gw_vlan)
    gw_reject = sort_ip_dict(ldict_gw_reject)

    print "\n"*2
    print "TMM IP Routes: (Name: Net/Mask -> Gateway IP)"
    for x in gw_ip:
            print "\t%s: %s/%s -> %s" % (x[0], x[2]['address'], x[2]['netmask'], x[3])

    print "\n"*2
    print "TMM Pool Routes: (Name: Net/Mask -> Gateway Pool)"
    for x in gw_pool:
            print "\t%s: %s/%s -> %s" % (x[0], x[2]['address'], x[2]['netmask'], x[3])

    print "\n"*2
    print "TMM Vlan Routes: (Name: Net/Mask -> Gateway Vlan)"
    for x in gw_vlan:
            print "\t%s: %s/%s -> %s" % (x[0], x[2]['address'], x[2]['netmask'], x[3])

    print "\n"*2
    print "TMM Rejected Routes: (Name: Net/Mask)"
    for x in gw_reject:
            print "\t%s: %s/%s" % (x[0], x[2]['address'], x[2]['netmask'])
```

The Route File Formats

When adding/removing routes, the following file formats are necessary to work with the script.

```
### Add Routes File Format ###
name,address,netmask,route_type,gateway
/Common/r5,172.16.1.0,255.255.255.0,pool,/Common/testpool
/Common/r6,172.16.2.0,255.255.255.0,vlan,/Common/vmnet3
/Common/r7,172.16.3.0,255.255.255.0,gateway,10.10.10.1
/Common/r8,172.16.4.0,255.255.255.0,gateway,10.10.10.1

### Delete Routes File Format ###
destination,netmask
172.16.1.0,255.255.255.0
172.16.2.0,255.255.255.0
172.16.3.0,255.255.255.0
172.16.4.0,255.255.255.0
```

The Test

Now that the script and the test files are prepared, let's take this for a spin! First, I'll run this without file arguments.

```
C:\>python getRoutes.py -s 192.168.6.5 -u admin
admin, enter your Password:

TMM IP Routes: (Name: Net/Mask -> Gateway IP)
        /Common/r.default: 0.0.0.0/0.0.0.0 -> 10.10.10.1
        /Common/r1: 65.23.5.88/255.255.255.248 -> 10.10.10.1
        /Common/r2: 192.32.32.0/255.255.255.0 -> 10.10.10.1

TMM Pool Routes: (Name: Net/Mask -> Gateway Pool)

TMM Vlan Routes: (Name: Net/Mask -> Gateway Vlan)

TMM Rejected Routes: (Name: Net/Mask)
```

Now, I'll add some routes (same as shown above in the formats section.)

```
C:\>python getRoutes.py -s 192.168.6.5 -u admin -a routes
admin, enter your Password:

TMM IP Routes: (Name: Net/Mask -> Gateway IP)
        /Common/r.default: 0.0.0.0/0.0.0.0 -> 10.10.10.1
        /Common/r1: 65.23.5.88/255.255.255.248 -> 10.10.10.1
        /Common/r7: 172.16.3.0/255.255.255.0 -> 10.10.10.1
        /Common/r8: 172.16.4.0/255.255.255.0 -> 10.10.10.1
        /Common/r2: 192.32.32.0/255.255.255.0 -> 10.10.10.1

TMM Pool Routes: (Name: Net/Mask -> Gateway Pool)
        /Common/r5: 172.16.1.0/255.255.255.0 -> /Common/testpool

TMM Vlan Routes: (Name: Net/Mask -> Gateway Vlan)
        /Common/r6: 172.16.2.0/255.255.255.0 -> /Common/vmnet3

TMM Rejected Routes: (Name: Net/Mask)
```

You can see that routes r5-r8 were added to the system. Now, I'll delete them.

```
C:\>python getRoutes.py -s 192.168.6.5 -u admin -d routedel
admin, enter your Password:
 Deleting Route 172.16.1.0/255.255.255.0
Deleting Route 172.16.2.0/255.255.255.0
Deleting Route 172.16.3.0/255.255.255.0
Deleting Route 172.16.4.0/255.255.255.0

TMM IP Routes: (Name: Net/Mask -> Gateway IP)
        /Common/r.default: 0.0.0.0/0.0.0.0 -> 10.10.10.1
        /Common/r1: 65.23.5.88/255.255.255.248 -> 10.10.10.1
        /Common/r2: 192.32.32.0/255.255.255.0 -> 10.10.10.1

TMM Pool Routes: (Name: Net/Mask -> Gateway Pool)

TMM Vlan Routes: (Name: Net/Mask -> Gateway Vlan)
```

```
TMM Vlan Routes: (Name: Net/Mask -> Gateway Vlan)

TMM Rejected Routes: (Name: Net/Mask)
```

Conclusion

Hopefully this was a useful exercise in converting pycontrol code to bigsuds. Looks like i have my work cut out for me in converting the rest of the codeshare! This example in full is in the iControl codeshare.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com