Some Snippets of Python in EXOS

(Compiled by M.Helm)

Thanks to Dave Hammers, Lance Richardson, & Jake Garver!

Script vs. Application

- exos.api can only be imported in a python application process
- exsh can only be imported in a python script
- This can be used to allow a .py file to be run in either context without error

```
try:
    import exos.api
    iAmProcess = True
except:
    iAmProcess = None

try:
    import exsh
    iAmScript = True
except:
    iAmScript = None
…
if iAmProcess:
   reply = exos.api.exec_cli([str(cmd)], ignore_errors=True)
if iAmScript:
   reply = exsh.clicmd(cmd, True)
```

show output vs. debug output

- In a python application a CLI command typically is executed and output returned using the `exos.api.exec_cli()` method.
- Debug commands, however, do not return output via this method. Instead, the following needs to be used:

```
from subprocess import check_output
reply = check_output(['/exos/bin/exsh','-n','0','-b','-c',cmd])
```

- This instantiates a separate EXOS shell, passes the command and collects the output.

Interaction in EXOS Python Shell

- From an ordinary EXOS python shell, one cannot import exsh (in order to execute EXOS commands and collect output):

```
import os,sys
os.system("sh")
* x201.6 # load script sh


BusyBox v1.13.4 (2015-10-06 19:06:46 EDT) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/exos/bin $ python
Python 2.7.3 (default, Mar 10 2014, 13:45:26)
[GCC 4.7.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import exsh
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named exsh
```

Interaction in EXOS Python Shell

- Use the `code.interact()` method to enter a Python shell that allows the import of exsh for easier development:

```
* x201.7 # load script cat py.py
import code
code.interact()
* x201.8 # load script py
Python 2.7.3 (default, Mar 10 2014, 13:45:26)
[GCC 4.7.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> import exsh
>>> s = exsh.clicmd("show switch", True)
>>> sl = s.splitlines()
>>> sl
['', 'SysName:          x201', 'SysLocation:      ', 'SysContact:       support@extremenetworks.com, +1
888 257 3000', 'System MAC:       08:00:27:BD:1A:A4', 'System Type:      Summit-PC', '', 'SysHealth
check:  Enabled', 'Recovery Mode:    All', 'System Watchdog:  Enabled', '', 'Current Time:     Sat Sep
19 12:16:51 2015', 'Timezone:         [Auto DST Disabled] GMT Offset: 0 minutes, name is UTC.', 'Boot
Time:        Fri Sep 18 06:59:12 2015', 'Boot Count:       0', 'Next Reboot:      None scheduled',
'System UpTime:    1 day 5 hours 17 minutes 39 seconds ', '', 'Current State:    OPERATIONAL
', 'Image Selected:   secondary              ', 'Image Booted:     secondary              ', 'Primary
ver:      16.1.1.4               ', 'Secondary ver:    16.1.2.14   ', '', 'Config Selected:
primary.cfg                                          ', 'Config Booted:    default.xsf
', '', 'primary.cfg       Created by ExtremeXOS version 16.1.2.14', '                 145801 bytes
saved on Fri Sep 18 06:59:25 2015']
```

Finding the first item in a list

- Example using `enumerate()` and `re.search()`

```
* x201.15 # load script cat example.py
import re
import exsh
s = exsh.clicmd("show switch", True)
sl = s.splitlines()
ix = next ((i for i, item in enumerate(sl) if re.search('.*System MAC.*', item)), -1)
l = sl[ix]
print l
ln = l.split()
MAC = ln[2]
print MAC
* x201.16 # load script example
System MAC:       08:00:27:BD:1A:A4
08:00:27:BD:1A:A4
```

Creating a log message

```
* x201.19 # load script cat example.py
m = "Hello World!"
cmd = 'create log message "{0}"'.format(m)
reply = exsh.clicmd(cmd, True)

* x201.20 # clear log
* x201.21 # load script example
* x201.22 # show log
09/19/2015 12:35:41.34 <Info:System.userComment> : Hello World!

A total of 1 log messages were displayed
```

Creating and Writing to a File

```
* x201.26 # load script cat example.py
import os
import sys

try:
   fp = open('/config/outfile.txt', 'w')
   fp.write('This is a line in a file.\n')
   fp.close()
except:
   print "Can't open file."

* x201.27 # load script example
* x201.28 # load script cat outfile.txt
this is a line in a file
```

Log Event Based Triggering Example:

```
import subprocess
import re

# start an EXOS shell process
p = subprocess.Popen(['/exos/bin/exsh','-n0', '-b', '-e','remote_serial'],
    stdin = subprocess.PIPE,
    stdout = subprocess.PIPE,
    stderr = subprocess.PIPE)

# configure it to be a log target with filtered messages
print >> p.stdin, 'create log filter t'
print >> p.stdin, 'config log filter t add events "System.userComment"'
print >> p.stdin, 'config log target session filter t'
print >> p.stdin, 'enable log target session'

# pre-compile a date seach for regex
date_patern = re.compile('\d\d\/\d\d\/\d\d\d\d')


while True:
    # read the output from the shell that is capturing the filtered log messages
    buf = p.stdout.readline().strip()

    # check if the shell went away for some reason
    if buf == '' and p.poll() is not None:
        break

    # we've got a buffer of data from the shell
    if buf:
        # search for the date that starts the log entry
        m = date_patern.search(buf)
        if m is None:
            continue
        # do something with the found string
        log_msg = buf[m.start():]
        print log_msg
```

Debug commands w/out Debug Mode

- While this must be run (and then deleted) as a process, this bypasses the need for answering the debug-mode challenge. This example runs TFTPdump.

```
import subprocess

p = subprocess.Popen(['/exos/bin/exsh','-n0', '-b', '-d', '-e','remote_serial'],
        stdin = subprocess.PIPE,
        stdout = subprocess.PIPE,
        stderr = subprocess.PIPE)

print >> p.stdin, '!tcpdump -n -i Broadcom -c 100 -w /config/out.pcap'
```

A python process that can send commands to an EXOS shell (in debug mode) and print output to the serial console and to file session.txt

```
import subprocess
import re

p = subprocess.Popen(['/exos/bin/exsh','-n0', '-b', '-d', '-e','remote_serial'],
    stdin = subprocess.PIPE,
    stdout = subprocess.PIPE,
    stderr = subprocess.PIPE)

print >> p.stdin, 'create log filter t'
print >> p.stdin, 'config log filter t add events "System.userComment"'
print >> p.stdin, 'config log target session filter t'
print >> p.stdin, 'enable log target session'
print >> p.stdin, 'config log target session match COMMAND->'
print >> p.stdin, 'disable clip'

fp = open('/config/session.txt', 'a')

while True:
    buf = p.stdout.readline().strip()

    if buf == '' and p.poll() is not None:
        break

    if buf:
        m = re.match('.+COMMAND\-\>(.*)', buf)
        if m is None:
            print buf
            print >> fp, buf
            fp.flush()
            continue
        cmd = m.group(1)
        print >> p.stdin, cmd
```

Example command from CLI: **`create log message "COMMAND->!tcpdump -n -i Broadcom -c 100 -w /config/out.pcap"`**

Using pexpect & SSHing to neighboring devices w/ UPM to restart a process.

```
#vi test.py
import pexpect
import exos.api
def exosCmd(cmd):
   reply = exos.api.exec_cli([str(cmd)], ignore_errors=True)
   return str(reply)

def logMsg(m):
    exosCmd('create log message "{0}"'.format(m))

logMsg("Starting remote save process!")
p = pexpect.spawn('/exos/bin/ssh -r 2 admin@10.0.0.202')
idx = p.expect(['password', pexpect.EOF, pexpect.TIMEOUT])
p.sendline('admin')
idx = p.expect(['x202', pexpect.EOF, pexpect.TIMEOUT])
p.sendline('create log message "test test test"')
idx = p.expect(['x202', pexpect.EOF, pexpect.TIMEOUT])
p.sendline('disable cli prompt')
idx = p.expect(['x202', pexpect.EOF, pexpect.TIMEOUT])
p.sendline('save')
idx = p.expect(['x202', pexpect.EOF, pexpect.TIMEOUT])
p.sendline('exit')

#^wq!

create process test python test start auto

create upm profile test
disable cli prompt
restart process test
.
create upm timer test
configure upm timer test profile test
configure upm timer test after 1 every 300
```

Augmenting EXOS

We await an 'alias' capability!

- CAT -

```
#cat.py
import sys, os
a = sys.argv[1]
cmd = "cat /config/"+a
os.system(cmd)
```

- CLEAR -

```
#cls.py
import os, sys
os.system('clear')
```

- VI (full) -

```
#vi.py
import sys, os
a = sys.argv[1]
cmd = "vi /config/"+a
os.system(cmd)
```

- MORE -

```
#more.py
import sys, os
a = sys.argv[1]
cmd = "more /config/"+a
os.system(cmd)
```