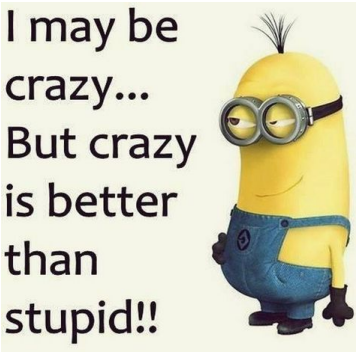
XMC 8.5 Workshop
XMC Advanced coding (for experts only)

Markus Nikulski
Sr. Corporate System Engineer

October 2020

Extreme

Disclaimer



Only for experts who know what they are doing!

Everything shown in this slide deck is not officially supported!
Opening a GTAC case is not an option!

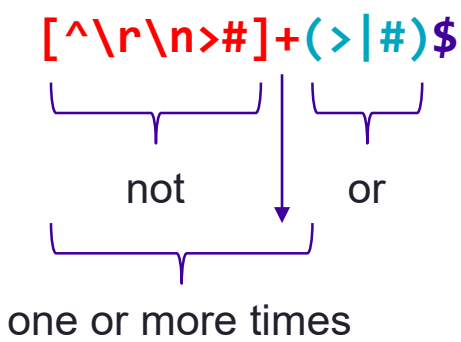
An XMC upgrade can may remove, overwrite or unregister modifications!

Device CLI prompt handling

for experts only

CLI prompt handling (the challenge)

```
OOB-R12>enable
OOB-R12#show vlan
Id    Name                   Type      Protocol          PID      Active IVL/SVL Mgmt
----  ---------------------- --------  ----------------  -------  ------ -------  ----
1     OOB                    Port      None              0x0000   Yes    IVL      Yes
         Port Members: 1/1-30
Total VLANs: 1
OOB-R12#
```

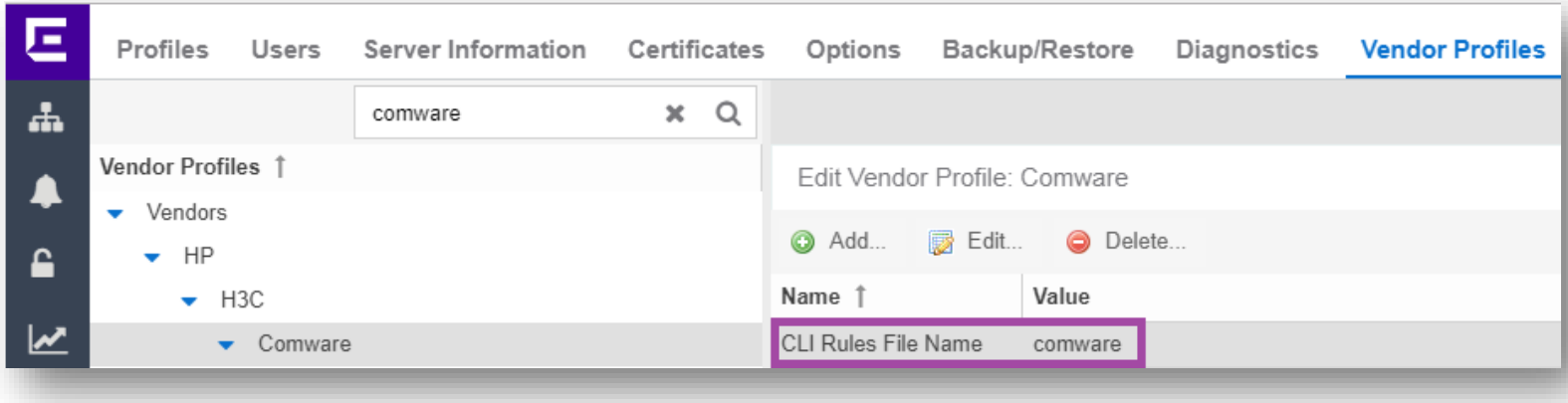


[ ]	group of possible characters
+	quantifier
\r	carriage return (0x0D)
\n	line feed (0x0A)
( )	group
|	‘or’ expression of character(s)
$	datastream end

CLI prompt handling

Is still in beta

Some devices response with an unexpected prompt. For such devices you can create **appdata/scripting/myCLIRules.xml** (no reboot is required).
The myCLIRules.xml "Rule name" must match the Vendor Profiles -> CLI Rules File Name variable.
Vendor Profiles are inherited; you can configure the variable for an entire device-family.



CLI prompt handling

myCLIRules.xml

```
<CLIRules>
    <Rule name="comware">
            <ShellPrompt>
                <defaultPrompt>
                  <prompt>[>\]]$</prompt>
                </defaultPrompt>
            </ShellPrompt>
            <LoginPrompt>
              <defaultPrompt>
                <prompt>(?i)login:</prompt>
              </defaultPrompt>
            </LoginPrompt>
            <PasswordPrompt>
               <defaultPrompt>
                 <prompt>(?i)^(password):</prompt>
               </defaultPrompt>
            </PasswordPrompt>
            <CommandPrompt command=".*">
                <defaultPrompt>
                    <prompt>Press|to continue or|to quit:</prompt>
                    <reply>y</reply>
                </defaultPrompt>
                <errors>
                    <messagePattern>$ERROR</messagePattern>
                </errors>
            </CommandPrompt>
    </Rule>
</CLIRules>
```

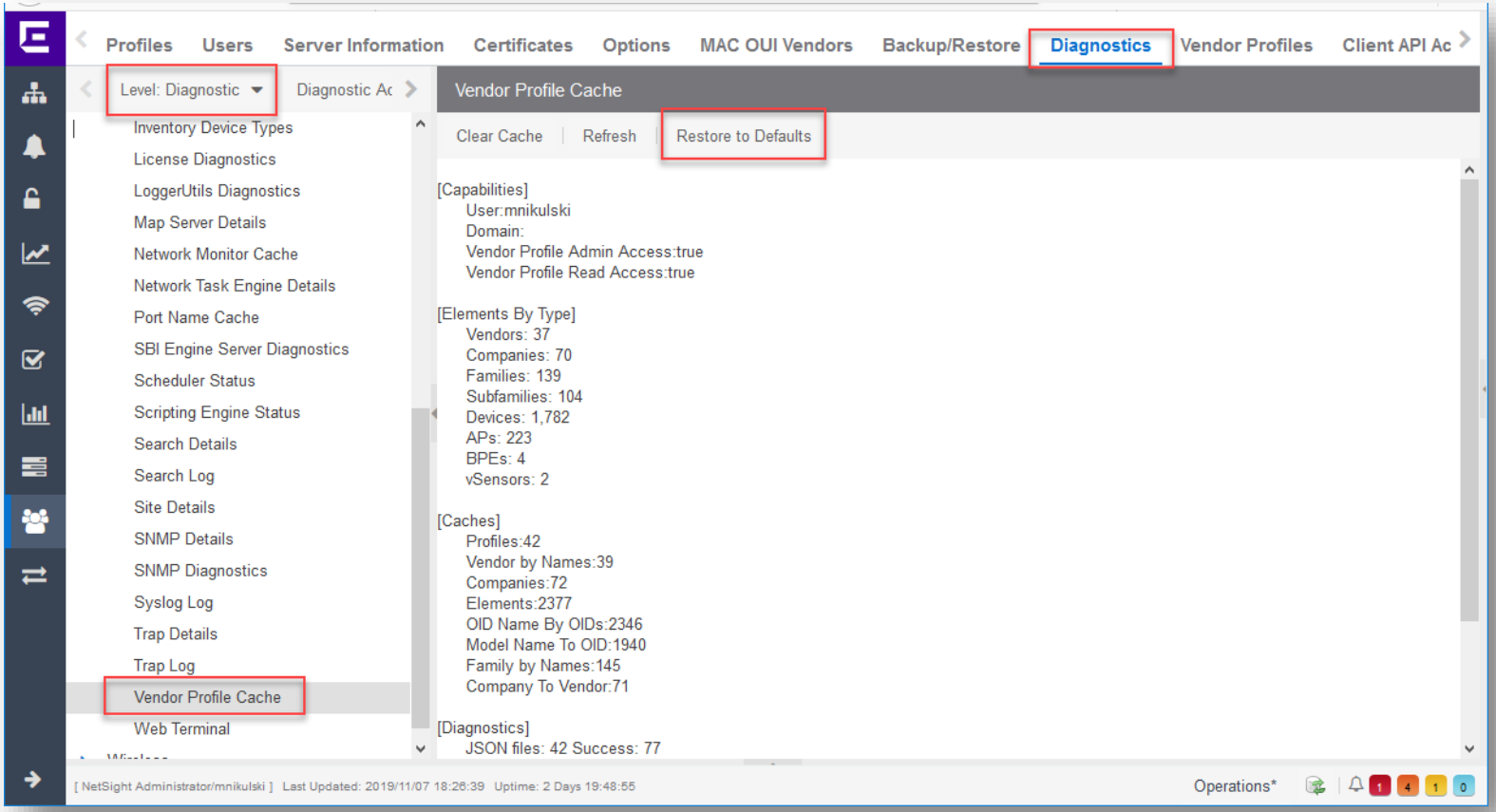
CLI prompt handling

CLI output ending with

```
<CLIRules>
    <Rule name="comware">
        <ShellPrompt>
            <defaultPrompt>
              <prompt>[>\]]$</prompt>
            </defaultPrompt>
        </ShellPrompt>
        <LoginPrompt>
          <defaultPrompt>
            <prompt>(?i)login:</prompt>
          </defaultPrompt>
        </LoginPrompt>
```

Using regular expressions	match
`[>\]]$`	`>$` or `]$`
`(?i)login:`	`login` or `Login`

reset Vendor profile manipulations



Python Package Manager

for experts only

activate PIP on XMC

```
cd /usr/local/Extreme_Networks/NetSight/jython/bin

export JAVA_HOME=/usr/local/Extreme_Networks/NetSight/java

sudo chmod a+x jython
sudo chmod a+x pip

./pip install ipaddress
```

Not all Python packages will work under Jython!

If modules used they should reported back to Extreme so we can consider adding them by default.

activate PIP on XMC

```
root@xmc.reading.ctc.local:~$ cd /usr/local/Extreme_Networks/NetSight/jython/bin
root@xmc.reading.ctc.local:/usr/local/Extreme_Networks/NetSight/jython/bin$ export JAVA_HOME=/usr/local/Extreme_Networks/NetSight/java
root@xmc.reading.ctc.local:/usr/local/Extreme_Networks/NetSight/jython/bin$ sudo chmod a+x pip
root@xmc.reading.ctc.local:/usr/local/Extreme_Networks/NetSight/jython/bin$ ./pip install ipaddress
Collecting ipaddress
  Downloading https://files.pythonhosted.org/packages/c2/f8/49697181b1651d8347d24c095ce46c7346c37335ddc7d255833e7cde674d/ipaddress-
1.0.23-py2.py3-none-any.whl
Installing collected packages: ipaddress
Successfully installed ipaddress-1.0.23
You are using pip version 9.0.1, however version 20.0.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
root@xmc.reading.ctc.local:/usr/local/Extreme_Networks/NetSight/jython/bin$
```

do not upgrade PIP!

Using PIP

```
pip --version

pip help

pip list
pip list --outdated
pip show <package-name>
pip show --files <package-name>

pip search <package-name>

pip install <package-name>
pip install --upgrade <package-name>

pip uninstall <package-name>
```


using your own XMC Python class

for experts only

Jython class path

Own Python classes should go under **appdata/scripting/extensions**

```
root@xmc.reading.ctc.local:~$ mkdir -p /usr/local/Extreme_Networks/NetSight/appdata/scripting/extensions
root@xmc.reading.ctc.local:~$ echo "
class MyFirstClass:
    def __init__(self, name):
        self.data = 'my name is ' + name

    def getText(self):
        return self.data
" > /usr/local/Extreme_Networks/NetSight/appdata/scripting/extensions/MyClass.py
root@xmc.reading.ctc.local:/usr/local/Extreme_Networks/NetSight/appdata/scripting/system/extensions$
```

NOTE: Class files are not part of the XMC backup!

own class execution

Edit Script: Test

Overview | Content | Description | Runtime Setti

```
1 import MyClass
2
3 test = MyClass.MyFirstClass('Max')
4
5 print test.getText()
6
```

MyClass.py

```
class MyFirstClass:
    def __init__(self, name):
        self.data =  'my name is ' + name

    def getText(self):
        return self.data
```

Run Script: Test

1. Device Selection | 2. Device Settings | 3. Runtime Settings | 4. Verify Run Script | 5. Results

Script Information

Task Information: Run Now
Script Name: Test
Script Task Name: N/A
Timeout (sec): 60

Overall Status

COMPLETED

Devices

Name	IP Address	Start Time/Total Run Time
10.128.0.0/20	10.253.0.16	2/24/2020 10:10:42 AM/(0 sec)

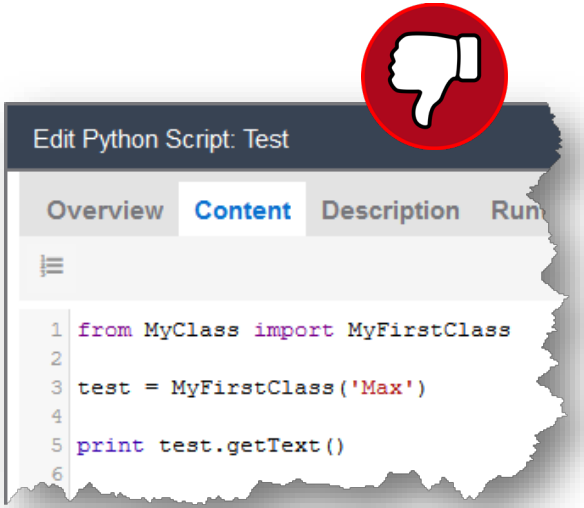
Results

```
Script Name: Test
Date and Time: 2020-02-24T09:10:42.512
XMC User: mnikulski
XMC User Domain:
IP: 10.253.0.16
my name is Max
```

« Previous | Run | Close

```
root@xmc.reading.ctc.local:~$ ls -l /usr/local/Extreme_Networks/NetSight/appdata/scripting/extensions
total 8
-rw-r--r-- 1 root root  142 Feb 24 09:10  MyClass.py
-rw-r--r-- 1 root root 3686 Feb 24 10:08 'MyClass$py.class'
root@xmc.reading.ctc.local:~$
```

own class execution



avoid name collision

XMC Python using SNMP

for experts only

using SNMP-GET

```
from xmclib.snmplib import SnmpRequest
from xmclib.snmplib import SnmpVarbind


# create SNMP request object for the given IP
snmp_request = SnmpRequest( emc_vars["deviceIP"] )

# declare to query OID(s)
varbinds = [
            SnmpVarbind(
                        oid = "rcSysLastRunTimeConfigSave.0 "
                       )
          ]



# perform SNMP-GET
response = snmp_request.snmp_get( varbinds, timeout = 15 )

if response.ok:
    print "last configuration save: ",  response.vars[0].val
else:
    print  "ERROR: SNMP-GET fails"
```


using SNMP-SET

```
from xmclib.snmplib import SnmpRequest
from xmclib.snmplib import SnmpVarbind
from xmclib.snmplib import ASN_INTEGER

# create SNMP request object for the given IP
snmp_request = SnmpRequest( emc_vars["deviceIP"] )

# declare to set OID(s) with definition type and value
varbinds = [
            SnmpVarbind(
                        oid      = "rc2kBootConfigEnableSpbmConfigMode.1",
                        asn_type = ASN_INTEGER,
                        value    = "1"
                       )
          ]

# perform SNMP-SET
response = snmp_request.snmp_set( varbinds, timeout = 15 )

if response.ok:
    print "Enable SPBM boot config flag : success"
else:
    print "Enable SPBM boot config flag : failed"
```

using SNMP-SET

```
from xmclib.snmplib import SnmpRequest
from xmclib.snmplib import SnmpVarbind
from xmclib.snmplib import ASN_INTEGER
```

```
def ifAdminStatusDown(ipAddress, portIndex):
    snmp_request = SnmpRequest( ipAddress )

    try:
        # set the ifAdminStatus OID for the portIndex
        varbinds = [SnmpVarbind(
                        oid      = "1.3.6.1.2.1.2.2.1.7." + str( portIndex ),
                        asn_type = ASN_INTEGER,
                        value    = "2"
                    )]

        response = snmp_request.snmp_set(varbinds, timeout = 5 )

        if response and response.ok:
             print "Port set to down %s %s" % (ipAddress, portIndex)
            return True
        else:
            raise Exception("Unable to set port down %s %s" % (ipAddress, portIndex))
            response.dump()
            return False

    except Exception as e:
        print "ERROR" + e.message
        return False
```

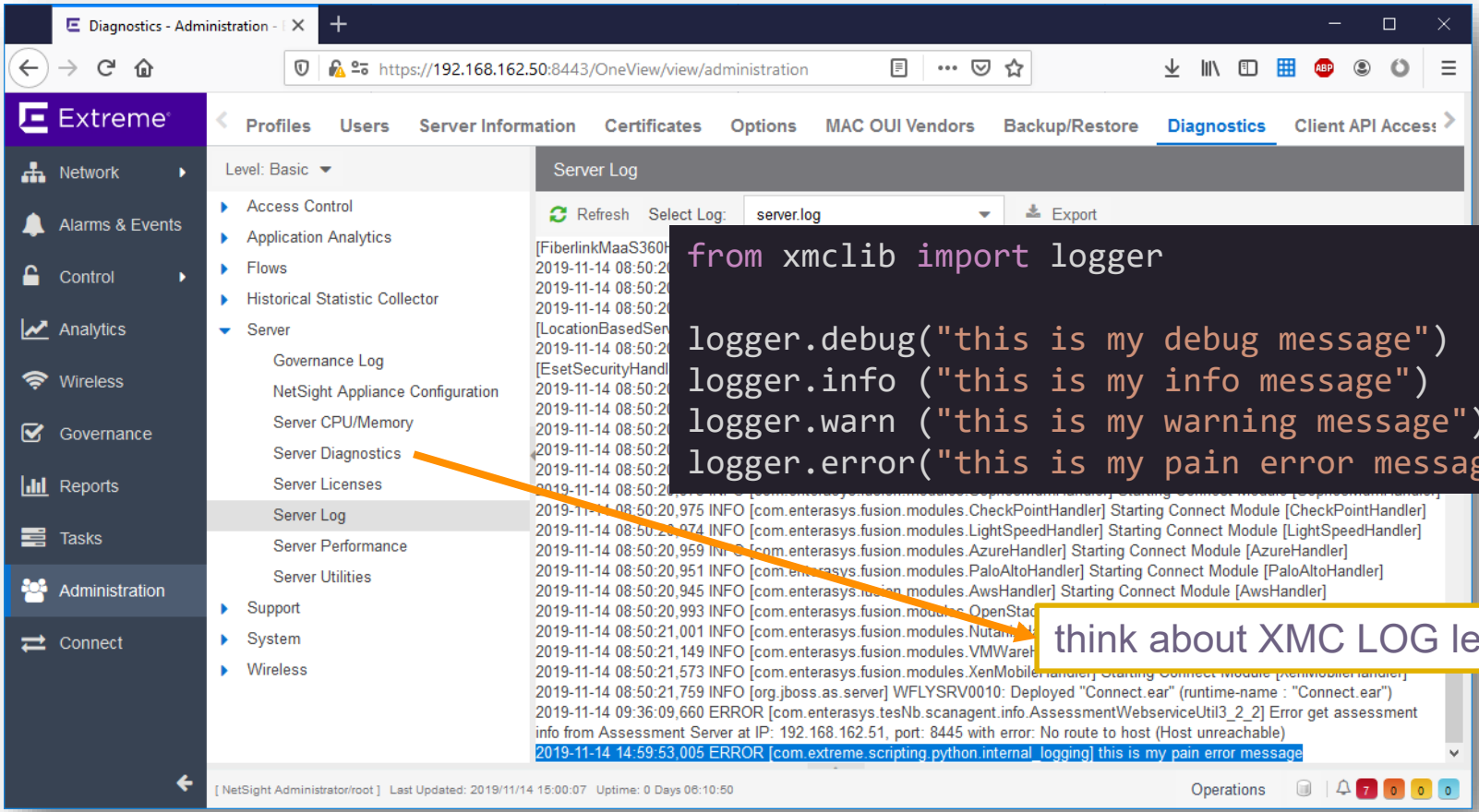
```
ifAdminStatusDown( "1.2.3.4", 7 )
```


write messages to the internal LOG facility

not recommended

internal logging

is not recommended



think about XMC LOG level !

Debugging

better to write your own log file

```
1  import os
2  import time
3  import logging
4
5  loggFileName  = emc_vars['deviceIP'].replace('.','_') + '-' + time.strftime("%Y%m%d-%H%M%S") + '.txt'
6  loggDirectory = str( os.path.abspath( os.path.join( emc_vars['jboss.server.log.dir'] + '/' )))
7
8  logging.basicConfig(level    = logging.DEBUG,
9                      format   = '%(asctime)s,%(msecs)-3d %(levelname)-8s [%(filename)s:%(lineno)d] %(message)s',
10                     filename = loggDirectory + loggFileName,
11                     filemode = 'w')
12
13 logging.debug('This message should go to the log file')
14 logging.info('So it should be')
15 logging.warning('And this, too')
16 logging.error('this is not good')
17 logging.critical('even worse')
```

executed LOG level

overwrite the file
'w+' will append

10-0-8-11_20191112-134855.txt

```
2019-11-13:12:16:53,470 DEBUG    [test.py:13] This message should go to the log file
2019-11-13:12:16:53,470 INFO     [test.py:14] So it should be
2019-11-13:12:16:53,470 WARNING  [test.py:15] And this, too
2019-11-13:12:16:53,471 ERROR    [test.py:16] this is not good
2019-11-13:12:16:53,471 CRITICAL [test.py:17] even worse
```


using the internal API (**WildFly**)

not recommended

using the internal API

```
from xmclib import cli
from device import api

message = emc_vars["banner"]

if api.enable() and api.config():
    cli.check_send("banner custom")
    cli.check_send("banner \"" + message + "\"")

    cli_result = api.save_config()
```

```
cli_result = api.reboot()

if cli_result and cli_result.isSuccess():
    restartStarted = True
```


using the internal API

```
from xmclib import cli
from re import match

if api.enable():
    cli_output = cli.send("show ip ssh  | include Administrative")

    if cli_output.isSuccess():
        isEnabled = cli.check_output_expr_list(cli_output, ["(?i)enabled?"])
        if isEnabled is None:
            print "SSH is enabled"
    else:
        print "SSH is disenabled"
```


using the internal API

```
cli_result = api.download_config(transferProtocol = emc_vars["transferProtocol"],
                                 serverAddress    = emc_vars["serverAddress"],
                                 serverPort       = emc_vars["serverPort"],
                                 isAnonymousLogin = emc_vars["isAnonymousLogin"],
                                 serverUsername   = emc_vars["serverUsername"],
                                 serverPassword   = emc_vars["serverPassword"],
                                 rootDirectory    = emc_vars["rootDirectory"],
                                 serverFilePath   = download_path,
                                 downloadTimeout  = emc_vars["downloadTimeout"])

cli.process_result( cli_result, emc_results )
```

using the internal API

```
cli_result = api.download_firmware(transferProtocol   = emc_vars["transferProtocol"],
                                   serverAddress      = emc_vars["serverAddress"],
                                   serverPort         = emc_vars["serverPort"],
                                   rootDirectory      = emc_vars["rootDirectory"],
                                   firmwareDirectory  = emc_vars["firmwareDirectory"],
                                   isAnonymousLogin   = emc_vars["isAnonymousLogin"],
                                   serverUsername     = emc_vars["serverUsername"],
                                   serverPassword     = emc_vars["serverPassword"],
                                   firmwareName       = emc_vars["firmwareName"],
                                   downloadTimeout    = emc_vars["downloadTimeout"]);

cli.process_result( cli_result, emc_results )
```


WWW.EXTREMENETWORKS.COM