Discover2013

It's time to build a better enterprise.
Together.

HP IMC custom scripting

Extending IMC for fun and profit

Lindsay Hill, Aaron Paxson / June 12, 2013

Agenda

- Introductions
- Developing device adapters
- Adding custom functions to IMC
- eAPI walkthrough

Introductions

Lindsay Hill

@northlandboy - http://lkhill.com

Primoris New Zealand (HP Partner)

- Network Management Consultant
- Install, Configure and Support HP IMC for medium-sized Enterprises.
- CCIE, CISSP, RHCE

Aaron Paxson

@Neelixx - http://myteneo.net

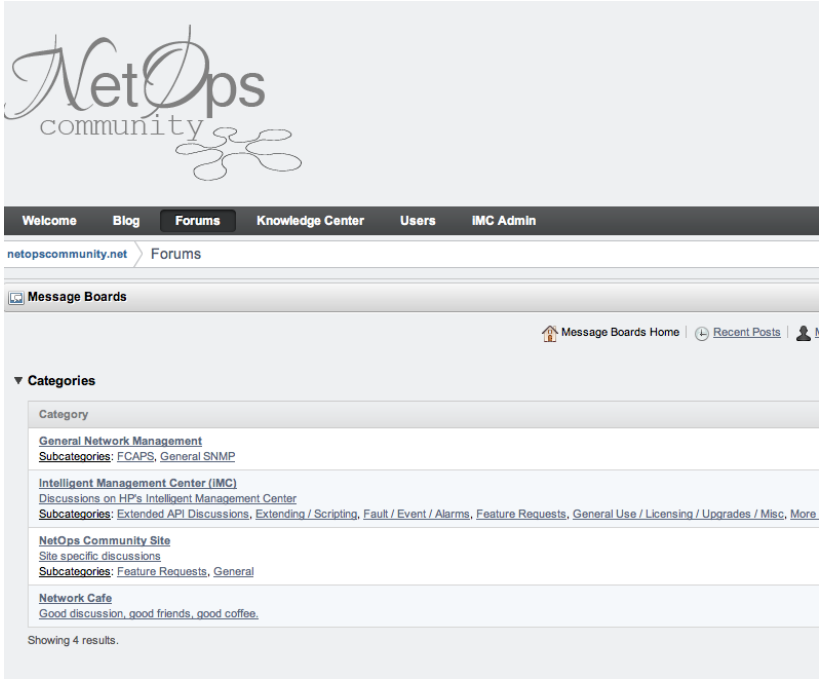
SVP Worldwide

- Global Network Manager
- HP IMC Customer

www.netopscommunity.net

Independent community forum, dedicated to helping fellow engineers with installing, using and extending IMC

GitHub repository of 3rd-party developed custom scripts and adapters – free to use



Building device adapters

Device adapters - overview

- IMC supports over 6,000 devices through the use of device adapters
- IMC has standardized functions and variables for configuration management – these cover config backup, config deployment and image deployment
- Adapters define how these functions are turned into device-specific commands, using XML, TCL and Perl
- Fully supported system for writing your own adapters to support new devices,or add custom functions to solve your business problems

Device adapter directories

Each vendor has its own directory with subdirectories for specific adapters

<IMC>/server/conf/adapters/ICC/

Cisco	Hewlett Packard	*<Vendor>*/adapter-index.xml - assigns sysOIDS to adapters
CiscoASA	HPGbE2C	*<Vendor>*/*<Adapter>*/adapter.xml describes services and points to other XML files used
CiscoCatNative	HPProLiant	
CiscoIOSGeneric	HPProcurve	
CiscoNX7K	HPProcurveMSM	
CiscoSNMP	HPProcurve3500	

Device adapters – files

XML files define capabilities while TCL/Expect files are used for running commands

adapter-index.xml

- Maps sysOIDs to specific adapters
- Can define default adapter

Adapter.xml

- Service definition for services offered by that adapter – common functions, backup, deployment
- Points to specific XML files

***_Builder.xml**

- Action definition
- Defines required actions, order, steps – maps to commands

***_Builder _Script.xml**

- Maps commands to individual TCL files
- Defines script prerequisites, and mode – e.g. CLI, TFTP

***.tcl**

- Carries out functions using Expect/TCL
- Parameters passed in
- CLI output parsed by Perl script

Device adapters – examples

Sample content from F5 adapter

F5/adapter-index.xml

```
<?xml version="1.0"?>
<!--sysoid adapt adapter-->
<adapters>
  <type name="CLI”>
    <adapter name="F5BIGIP">
            <description>F5 (multi-config) load-balancers, Big-IP series</description>
      <sysoid>1.3.6.1.4.1.3375.2.1.3.4.43</sysoid>
            <version series="F5BIG" vrp="" release=""/>
            <defaultver series="F5BIG" vrp="" release=""/>
    </adapter>
      </type>
</adapters>
```


Device adapters – examples

Sample content from F5 adapter (continued)

F5/F5BIGIP/adapter.xml

```
<adapter name="F5BIGIP">
    <description>F5 (multi-config) load-balancers, Big-IP series</description>
    <version>1.0.0</version>
    <services>
        <service name="CLICommon">
            <item type="common">F5_Common_CLI.xml</item>
        </service>
        <service name="ConfigBackup">
            <item type="builder_definition">F5_Config_Backup_Builder.xml</item>
            <item type="tcl_script">F5_Config_Backup_Builder_Script.xml</item>
        </service>
    </services>
</adapter>
```


Device adapters – examples

Sample content from F5 adapter (continued)

F5/F5BIGIP/F5_Config_Backup_Builder_Script.xml

```
<?xml version="1.0"?>
<scripts>
    <command name="backup_running_config_ftp" method="FTP">
        <error>Failed to upload configuration to FTP server.  FTP server may be down, or incorrectly
specified, command syntax may be incorrect, or prompts may not be what was expected.</error>
        <require-mode>exec</require-mode>
        <script>
    backup_running_config_ftp.tcl
        </script>
    </command>
</scripts>
```


Device adapters – examples

Sample content from F5 adapter (continued)

F5/F5BIGIP/backup_running_config_ftp.tcl

```
#*********************************************************************
# Identification:backup_running_config_ftp
# Purpose:      backup running config via ftp
#*********************************************************************

set timeout $very_long_timeout
set sourceFile "/var/local/ucs/imc_icc_f5_cfg.ucs"

ftp_trans_file true $sourceFile $TFTPFile

set timeout $standard_timeout
```


Quick recap: IMC backup methods

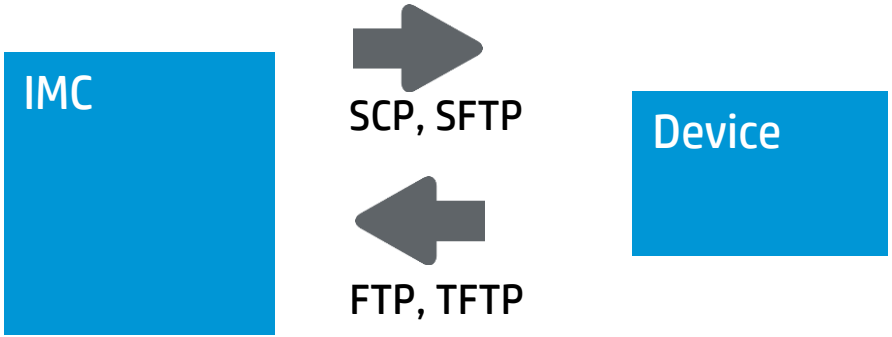
Key is understanding methods and data flow direction

File transfer methods:

- SNMP Read-Write and TFTP/FTP
- Telnet and TFTP/FTP
- SCP/SFTP
- CLI (Telnet/SSH) + “show run” (or equivalent)

Backup method selection

- IMC tries SNMP first if adapter defined
- Then tries file transfer method defined in configuration center -> options
- If that fails, falls back to CLI
- Can’t mix SNMP and SCP/SFTP



Build your own device adapter

Summary of steps involved

- Analyze your device:
  - What access methods does it support?
  - What commands do we need to run to take a backup? Is this similar to any other device already supported by IMC?
  - Does it have both a startup and a running configuration?
- Configure IMC:
  - Add device model, series and vendor to IMC. Note sysOID.
  - Create new folder <IMC>/server/conf/adapters/ICC/<Vendor>/
  - Create new adapter folder <IMC>/server/conf/adapters/ICC/<Vendor>/<Adapter_Name>
- Create adapter files – service definitions, and TCL files
- Restart IMC to pick up new adapter
- Discover device (or synchronize if already in IMC)
- Test!

Live walkthrough

Live walk through of process of adding new device adapter for Fortinet system

Will cover process of adding new device, creating adapter files, and running a backup

Shows logfiles to look at for troubleshooting

Troubleshooting process

- Check your credentials, login type, and file transfer mode
- Know what backup method you expect to be used (SCP, SFTP, CLI, TFTP, FTP)
- Manually run the sequence of backup commands first, from the IMC server
- Key log file is <IMC>/server/conf/log/imccfgbakdm*.log – tells you almost everything you need to know – but it takes some decoding!
- First check that the right adapter is being selected – look for lines like:
  - Device login type is 1,dev_id=4,**AdaptName=CiscoASA**
- Follow log file – see which TCL files are being executed, and what the results are
- Check <IMC>/server/tmp/ for temporary session files created during execution – these show all the output created. These are deleted on normal exit
- Use Wireshark to inspect traffic if required

Lessons from the field

- Copy and tweak an existing adapter if you can!
- Use unencrypted communications during development if possible – Wireshark can really help
- Set timeouts to low values during debugging – “set timeout 10”
- Keep it simple – start with just backups, using just one method. Add more later.
- Key locations:
  - Log file: <IMC>/server/conf/log/imccfgbakdm*.log
  - Temporary files: <IMC>/server/tmp/
  - Backups: <IMC>/server/data/cfgbak/
- Pay close attention to timestamps in imccfgbakdm log file – these may appear out of order
- Restart IMC and synchronize devices when first adding new adapters
- If you’re stuck, ask at NetOps: www.netopscommunity.net. We’ll do our best to help!

Adding custom functions

The overview

XML – a way to define data

95% of extending iMC with new devices is done in XML

Each tag has a start, and an end (i.e. <Start> </Start>)

Single tags can shortcut an end if no data (i.e. <Start />)

Tags can have parameters (i.e. <Start time="now" date="today" > </Start>)

Data is within tags:

```
<Start time="now" date="today">
  <action>Do It</action>
</Start>
```

Most XML files are loaded on startup. Changes require service restart.

Many XML files needed to define single process:

- Allows for delegation/control: You can define the adapter, and someone else can define the script actions, etc.
- Easier to modify: when extending a device, one large XML file is cumbersome.

Device adapters overview

The goal

Allow tier 1 / 2 level technicians to change allowed number of devices per port.

Use case: By default, all access switchports are set to 1. A user just received a new VoIP phone and hooked it up between the computer and network, but it will not connect

Interface - assign port security to Juniper EX switch

- Set number of Mac-addresses learned per port
- Assuming one desktop and one phone, we will set max to 2

The steps without IMC

What we need to do without using iMC

Interface - assign port security to Juniper EX switch

1. Access box through SSH/Telnet
2. Get into edit mode
3. “set ethernet-switching-options secure-access-port interface ge-0/0/1 mac-limit 2 action drop”
4. commit

What are the parameters?

Identify any data that we need to collect

Ask yourself, “What could change”?

1. “set interface ge-0/0/1 mac-limit 2 action drop”
   1. Which interface? - ge-0/0/1 (string showing interface name)
   2. How many to allow? - 2 (any integer)
   3. What to do if exceeded? - drop, log, none, shutdown

Input types

Text box

juniops_operations.xml

<Parameter name = “interfaceID” label = “Interface” required = “true”>

Interface: ge-0/0/22

Variable name = $interfaceID

Required field

Input types

Text box

juniops_operations.xml

<Parameter name = “maxAllowed” label=“Max Allowed” required = “true”>

Max Allowed: 2

Variable name = $maxAllowed

Required field

Input types

Drop-down list

juniops_operations.xml

```
<Parameter name = “portAction” label=“Resulting Action” required = “true”>
   <View type=“SelectOneMenu”>
      <SelectItem value=“drop” displayValue=“Drop frames”/>
      <SelectItem value=“log” displayValue=“Log attempts” />
      <SelectItem value=“none” displayValue=“Do nothing” />
      <SelectItem value=“shutdown” displayValue=“Shutdown Interface” />
   </View>
</Parameter>
```

Variable name = $portAction

Required field

The code

Developing scripts

Developing TCL/Expect scripts using Windows:

http://linux.about.com/od/softorther/a/Tcl-Expect-For-Windows-Linux-Interaction.htm

Expect user guide:

http://docs.activestate.com/activetcl/8.5/expect4win/ex_usage.html#code_tcl

Lessons from the field

- Copy and tweak an existing adapter if you can! Review and use existing TCL files.
- Make sure you have valid XML files. (i.e. terminated tags, correct syntax)
- Use unencrypted communications during development if possible – Wireshark can really help
  - Use this when questioning whether the right commands are being sent.
- Key locations:
  - Log file: <IMC>/server/conf/log/imccmddm*.log (sending commands)
  - Log file: <IMC>/client/log/imcforeground.log (for web interface)
  - Custom Web UI / Operations: <IMC>/client/web/apps/imc/gencfg/register/custom
  - Custom scripts/adapters: <IMC>/server/conf/adapters/custom
- Pay close attention to SYSOID mappings. This may answer why your device doesn't display when trying to select it.
- Restart IMC when adding new adapters. These get loaded on startup.
- No need to restart IMC when running scripts.
- If you're stuck, ask at NetOps: www.netopscommunity.net. We'll do our best to help!

eAPI code walkthrough

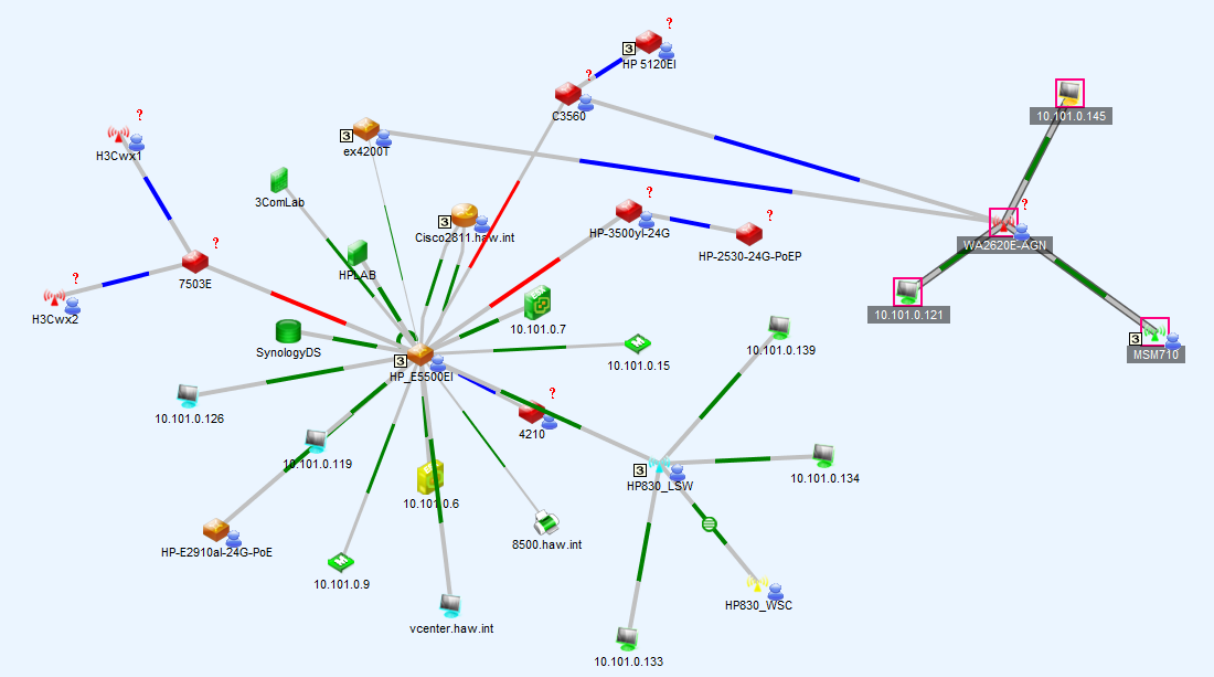
The business challenge

Health care

- Patient data needs to be protected
- Medical professionals need to care for patients
- Devices get misplaced

The technical challenge

Finding L2 addresses is a hop by hop scenario



Steps to a solution

1. Find the function in IMC
2. Find the eAPI
3. Write the code
4. Make it better

Find the function

Let’s take a look…

Using the real-time location service in IMC

Find the eAPI

eAPI documentation

Query Real-Time Locations

iMC Platform - Resource Manager

Query real-time locations with certain criteria.

Interface URI

/res/access/realtimeLocate

Parameters

Query parameters	
type	Location type. 1 for MAC address. 2 for IP address. Integer type. Required. The default value is 2.
value	Location address. String type. Required. The default value is an empty string.
total	Only the number of records that meet the requirements is returned. Boolean type. Optional. The default value is false. When the value is true, the returned message body is empty, and the Total attribute of the message header returns the number of records that meet the requirements.

Query the IMC eAPI services

http://imc_host:port/imcrs/application.wadl

Search for “real”

application.txt

rest_en.rar

RESTful test client

REST 测试客户端

协议： http 主机： 10.101.0.205 端口： 80 HTTP 方法： GET 用户名： admin 密码： •••••

URI： /imcrs/res/access/realtimeLocate?type=1&value=00-11-32-10-03-8b

请求消息头：

Key	Value
accept	application/xml

请求消息体：

发送

回应消息头：

```
Content-Type: application/xml
Content-Length: 258
Date: Sat, 27 Apr 2013 04:14:14 GMT
```

回应消息体：

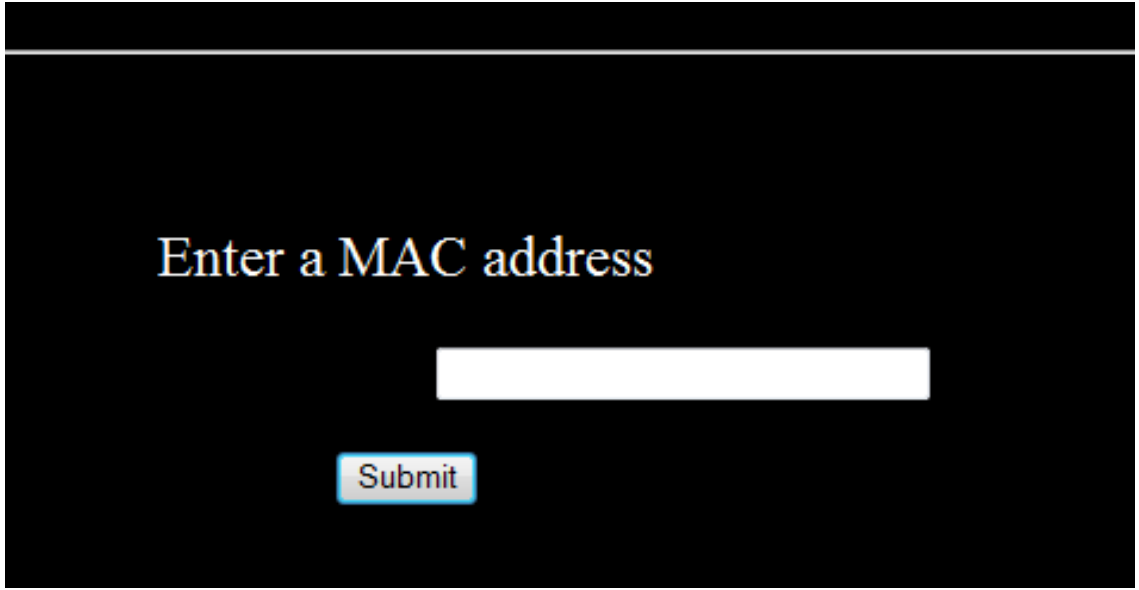
```
<?xml version="1.0" encoding="UTF-8"?>
<list>
  <realtimeLocation>
    <locateIp>00:11:32:10:03:8b</locateIp>
    <deviceId>41</deviceId>
    <deviceIp>10.101.0.221</deviceIp>
    <ifDesc>GigabitEthernet1/0/21</ifDesc>
    <ifIndex>21</ifIndex>
  </realtimeLocation>
</list>
```

eAPI real time location output

```
<?xml version="1.0" encoding="UTF-8"?>
<list>
 <realtimeLocation>
  <locateIp>00:11:32:10:03:8b</locateIp>          #MAC to be Located
  <deviceId>41</deviceId>                         #Device ID
  <deviceIp>10.101.0.221</deviceIp>               #Device IP
  <ifDesc>GigabitEthernet1/0/21</ifDesc>          #Interface Label
  <ifIndex>21</ifIndex>                           #Interface Index
 </realtimeLocation>
</list>
```


Write the code

Create an HTML form



macfind1htm

Getting the information from the webpage

```
#-----------------------------------------------------------------------
#          Get the field VARS from the calling HTML form
#-----------------------------------------------------------------------
form = cgi.FieldStorage()
MAC = form.getvalue("macaddress")
imc_url = '10.132.0.99'
L = MAC.split(',')
cr = "\n"
count = 0
# Get output file ready for writing
f = open('/var/www/pages/mac_find.txt','w')
a = "Mac Address"
b = "IP Address"
c = "Port"

report = "%s              %s   %s   " % (a,b,c)
```

columbo1.txt

Rename to columbo1.py

Setup the variables

```
#         Setup the URL VARS
#-----------------------------------------------------------------------
usernames='chewie'
passwords='chewie'
#-----------------------------------------------------------------------
#         Write report headers
#-----------------------------------------------------------------------
out = "=================================================================="
f.write(out)
f.write(cr)
out = "          MAC FIND                        wookieware 2013 "
f.write(out)
f.write(cr)
out = "=================================================================="
f.write(out)
f.write(cr)
f.write(report)
f.write(cr)
out = "------------------------------------------------------------------"
f.write(out)
f.write(cr)
```


Authenticate with IMC

```
#-----------------------------------------------------------------------
#          Define the AUTHORIZATION handler
#-----------------------------------------------------------------------

authhandler = urllib2.HTTPDigestAuthHandler()
authhandler.add_password("iMC RESTful Web Services", imc_url, usernames,
passwords)

#-----------------------------------------------------------------------
#          Authenticate with the IMC server
#-----------------------------------------------------------------------

opener = urllib2.build_opener(authhandler)
urllib2.install_opener(opener)
```

Send the eAPI string

```
#-----------------------------------------------------------------------
#          Send eAPI string to IMC server to get MAC
#-----------------------------------------------------------------------
for item in L:
        MACCAD = item.strip()
        count = count + 1
        mac_url='http://10.132.0.99/imcrs/res/access/realtimeLocate?type=1&value='
+MACCAD
        pagehandle=urllib2.Request(mac_url)
        pagehandle.add_header('Accept', 'application/xml')
```

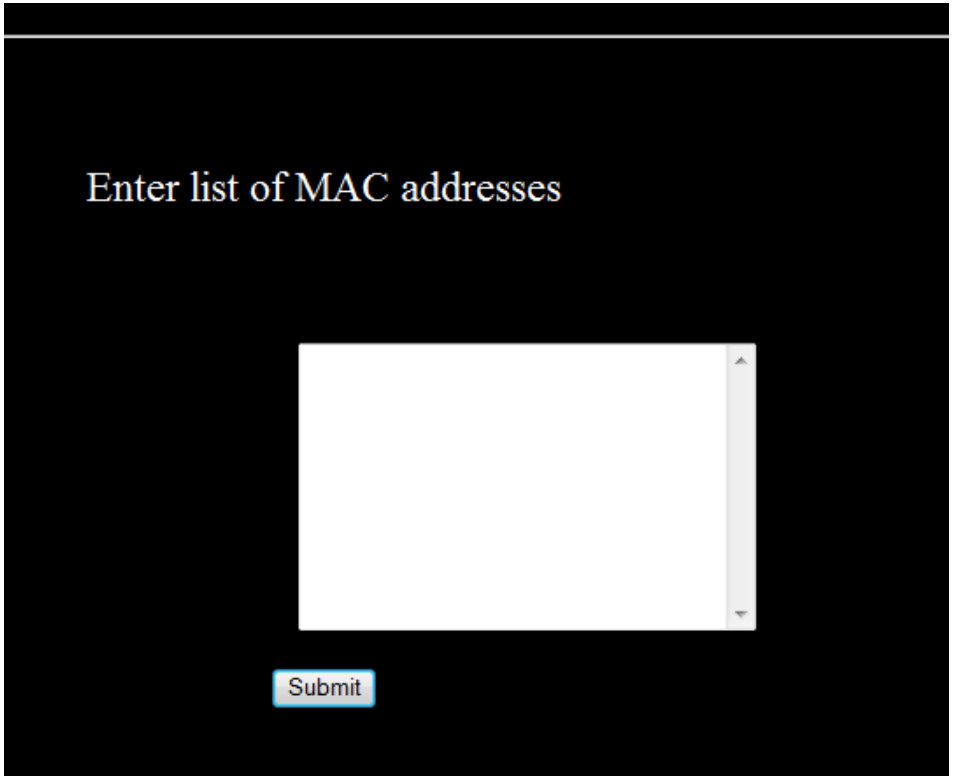

Decode the eAPI return

```
#-----------------------------------------------------------------------
#          Read from memory location addinfourl assign MAC "tree"
#-----------------------------------------------------------------------
        tree = xml.parse(result1)
        rootElement = tree.getroot()   #Gets the root of the element
# In case the DevIP could have more than one return. Get the first one only!
        for node in tree.iter():
                if (node.tag == 'deviceIp' and DevIP == 'null'):
                        DevIP = node.text          # DevIP
        for node in tree.iter():
                if node.tag == 'ifDesc':
                        DevINT = node.text
        line ="%s     %s     %s     " % (MACCAD,DevIP,DevINT)
# Write information to file
        f.write(line)
        DevIP = 'null'
# After process loop close file
f.close()
```

Let’s take a look…

Make it better

Create the web page - multiple MACs

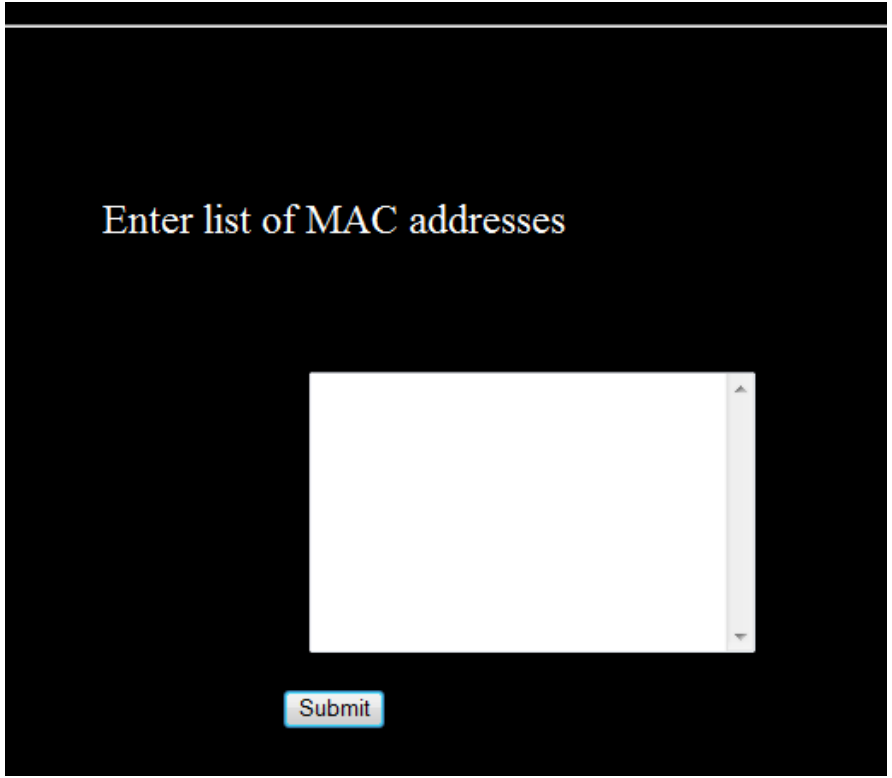


macfind2htm

Let’s take a look…

Make it better
Part 2

Create the web page – multiple Macs… plus!



macfind.htm

Enhance the script

1. Obtain device IP (DevIP) and interface description (DevIF)
2. Obtain device ID in IMC database (DevID)
3. Obtain device location (DevLoc)
4. Obtain device interface (DevINT)
5. Obtain interface name (DevName)

columbo0.txt

Find the MAC address

```
#-----------------------------------------------------------------------
#          Send eAPI string to IMC server to get MAC
#-----------------------------------------------------------------------
for item in L:
        MACCAD = item.strip()
# check for null value in MACCAD
        if MACCAD is None:
                f.close()
                sys.exit()
        count = count + 1
        mac_url='http://%s/imcrs/res/access/realtimeLocate?type=1&value=%s'%
(imc_url,MACCAD)
        pagehandle=urllib2.Request(mac_url)
        pagehandle.add_header('Accept', 'application/xml')
```

Find the device location

```
#-----------------------------------------------------------------------
#        Now create the IP URL and collect the DevLoc
#-----------------------------------------------------------------------
    ip_url='http://%s/imcrs/plat/res/device/%s'% (imc_url,DevID)
    pagehandle=urllib2.Request(ip_url)
    pagehandle.add_header('Accept', 'application/xml')
    result3 = urllib2.urlopen(pagehandle)
    tree = xml.parse(result3)
    for node in tree.iter():
        if node.tag == 'location':
            DevLoc = node.text

    for node in tree.iter():
        if node.tag == 'sysName':
            DevName = node.text
```

Find the interface Alias

```
#-----------------------------------------------------------------------
#         Create the IP URL and collect the ifAlias
##----------------------------------------------------------------------
#Change DevIF into interger
        intf = int(DevIF) - 1
        ip_url='http://%s/imcrs/plat/res/device/%s/interface?start=%s&size=1' %

(imc_url,DevID,intf)
#testing
        pagehandle=urllib2.Request(ip_url)
        pagehandle.add_header('Accept', 'application/xml')
        result4 = urllib2.urlopen(pagehandle)
        tree = xml.parse(result4)
        for node in tree.iter():
                if node.tag == 'ifAlias':
                        DevAlias = node.text
```


Let’s take a look…

For more information

Attend these sessions

- Session Id, name
- 11 pt. HP Simplified
- Session Id, name

Visit these demos

- Demo name, Demo number
- 11 pt. HP Simplified
- Demo name, Demo number

After the event

- Visit www.netopscommunity.net

Your feedback is important to us. Please take a few minutes to complete the session survey.

Learn more about this topic

Use HP Autonomy's Augmented Reality (AR) to access more content

1. Launch the **HP Autonomy AR** app*
2. View this slide through the app
3. Unlock additional information!

*Available on the App Store and Google Play

Thank you