Tech Note:
ClearPass Profiling
Version 1.2 July 2015

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Modified By</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>June 2014</td>
<td>Danny Jump</td>
<td>Initial Published Version 1.</td>
</tr>
<tr>
<td>1.1</td>
<td>October 2014</td>
<td>Danny Jump</td>
<td>Updated details for ActiveSync to add details of Exchange 2013 support.</td>
</tr>
<tr>
<td>1.2</td>
<td>July 2015</td>
<td>Danny Jump</td>
<td>Added details CPPM 6.5 profiling and Custom Device Classification</td>
</tr>
</tbody>
</table>
# Table of Contents

Overview .................................................................................................................. 6

ClearPass Profile ...................................................................................................... 6

Setup .......................................................................................................................... 6

Device Profile .......................................................................................................... 7

Collectors .................................................................................................................. 8

DHCP ......................................................................................................................... 8

ClearPass Onboard .................................................................................................. 12

HTTP User-Agent .................................................................................................... 12

MAC OUI ................................................................................................................... 12

ActiveSync plugin support for Exchange 2010 & 2013 ....................................... 12

CPPM OnGuard ....................................................................................................... 18

SNMP ......................................................................................................................... 18

Discovering endpoint with static IP address ....................................................... 20

Discovery via ARP Read .......................................................................................... 20

Discovery via Subnet Scanner .................................................................................. 21

One-Time Subnet Scans ........................................................................................... 22

IF-MAP ......................................................................................................................... 23

Cisco Device Sensor ............................................................................................... 25

Enterprise Mobility Management (EMM/MDM) .................................................... 27

TCP Fingerprinting ................................................................................................. 30

Profiling ..................................................................................................................... 32

Stage 1 ....................................................................................................................... 32

Stage 2 ....................................................................................................................... 33
Table of Figures

Figure 1 - Enabling 'Profiler' on a CPPM node................................................................. 6
Figure 2 - Adding multiple IP helpers on a switch/router.................................................. 8
Figure 3 - Configure Local Span port on older Cisco 2900/3500XL................................... 9
Figure 4 - Configuring Local SPAN port IOS 12.2(33) and later (not ALL Cisco switches) .... 9
Figure 5 - Configuring the RSPAN on 'Local' & 'Remote' Cisco switches .........................10
Figure 6 - Configuring the RSPAN monitor session on the 'Remote' switch ......................10
Figure 7 - Configuring the RSPAN monitor session on the 'Local' switch .........................11
Figure 8 - Configure SPAN on MAS..................................................................................11
Figure 9 - Error message to install .NET on Exchange 2013
Figure 10 - Installing .NET on server
Figure 11 - Enabling .NET after installation
Figure 12 - Activesync Plugin attributes in Endpoint
Figure 13 - Setting SNMP community attributes
Figure 14 – Setting ARP read frequency
Figure 15 - Setting community string and enabling ARP-read
Figure 16 - Assigning IP SUBNETS in Profiler to zones
Figure 17 - Configuring SUBNET scan frequency
Figure 18 - Defining On-Demand Subnet Scan
Figure 19 - On-Demand Subnet scan messages in Event Viewer
Figure 20 - Enabling Aruba Ctrl to send IF-MAP info to CPPM (GUI)
Figure 21 - Enabling Aruba Ctrl to send IF-MAP info to CPPM (CLI)
Figure 22 - Enabling device sensor on Cisco switch
Figure 23 - Configuring device sensor on Cisco switch
Figure 24 - Enabling device sensor LLDP TLV attributes
Figure 25 - Enabling device sensor CDP TLV attributes
Figure 26 - Enabling device sensor filter for DHCP, LLDP & CDP
Figure 27 – Globally enable LLDP
Figure 28 – Enable LLDP on an interface
Figure 29 – Globally enable CDP
Figure 30 - Example of EMM attributes #1
Figure 31 - Example of EMM attributes #2
Figure 32 - Example of EMM attributes #3
Figure 33 - Adding an MDM context server
Overview

The following guide has been produced to help educate our customers and partners in understanding ClearPass endpoint profiling.

**Note:** Where you see a red-chili this is to signify a ‘hot’ important point and highlights that this point is to be taken as a best-practice recommendation.

ClearPass Profile

Profile is a ClearPass module that automatically classifies endpoints using attributes obtained from software components called Collectors. As an example it can be used to implement BYOD flows where access has to be controlled based on the type of the device and the identity of the user. Profile can be set up in a network with minimal amount of configuration.

Setup

To classify devices using Profile, you need to set up the following:

Select one of the CPPM nodes in the Zone as profiler. Navigate to Administration » Server Manager » Server Configuration as shown below in Figure1.

![Figure 1 - Enabling 'Profiler' on a CPPM node](image-url)
Once devices are classified, you can use them in policies to control access in your network. You can use the `Authorization:[Endpoints Repository]` attributes in the CPPM Role OUI Mapping Policy. See section titled “Endpoint Profile Store as Authorization Source” for more information.

## Device Profile

A device profile is a hierarchical model consisting of 3 elements - **DeviceCategory**, **DeviceFamily**, and **DeviceName** derived by Profile from endpoint attributes.

- **DeviceCategory** – This is the broadest classification of a device. It denotes the type of the device. **Example:** Computer, Smartdevice, Printer, Access Point, etc.

- **DeviceFamily** – This element classifies devices into a category; this is organized based on the type of OS or type of vendor. **Example:** Windows, Linux, Mac OS X are some of the families when DeviceCategory is Computer. Apple, Android are examples of DeviceFamily when DeviceCategory is SmartDevice.

- **DeviceName** - Devices in a family are further organized based on more granular details such as version. **Example:** Windows 7, Windows 2008 server are device names under Windows family (DeviceFamily).

This hierarchical model provides a structured view of all endpoints accessing the network.

Apart from the these, Profile also collects and stores...

- IP Address
- Hostname
- MAC Vendor
- Timestamp when device was first discovered
- Timestamp when device was last seen
Collectors

Collectors are network elements that provide data to profile endpoints. The following collectors send endpoint attributes to Profile:

- DHCP
  - DHCP snooping
  - Span ports
- ClearPass Onboard
- HTTP User-Agent
- MAC OUI – Acquired via various auth mechanisms such as 802.1X, MAC auth, etc.
- ActiveSync plugin (Exchange 2010 & 2013)
- CPPM OnGuard
- SNMP
- Subnet Scanner
- IF-MAP
- Cisco Device Sensor (Radius Accounting)
- MDM
- TCP Fingerprinting

**DHCP**

DHCP attributes such as option55 (parameter request list), option60 (vendor class) and options list from DISCOVER and REQUEST packets can uniquely fingerprint most devices that use the DHCP mechanism to acquire an IP address on the network. Switches and controllers can be configured to forward DHCP packets such as DISCOVER, REQUEST and INFORM to CPPM (DHCP Relay/ IP-Helper). These DHCP packets are decoded by CPPM to arrive at the device category, family, and name. Apart from fingerprints, DHCP also provides hostname and IP address.

**DHCP Relay Agent – Aruba/Cisco**

Configuring Aruba Controller and Cisco Switch to Send DHCP Traffic to CPPM

```plaintext
interface <VLAN_NAME>
ip address <IP_ADDR> <NETMASK>
ip helper-address <DHCP SERVER IP>
ip helper-address <CPPM IP>
```

**Figure 2 - Adding multiple IP helpers on a switch/router**

Notice how multiple ‘ip helper-address’ can be configured to send DHCP packets to servers other than the DHCP server, i.e. a CPPM node.
**DHCP SPAN**

Certain networks precipitate the need to receive DHCP packets off a mirrored port, instead of relying on DHCP relays, which is used by CPPM for device profiling. In earlier release we support only dhcp relays. Starting in the CPPM 6.3 release we support SPAN for receiving DHCP packets. Currently only the 25K HW appliances has additional ports beyond the two MGMT/DATA interfaces where this can be utilized as a dedicated interface.

**SPAN Configuration:**

SPAN Port Configuration has to be done on switches where DHCP Servers (Source) and CPPM Servers (Destination) are connected.

**Cisco Switch SPAN Configuration:**

**Local SPAN:** Mirrors traffic from one or more interface on the switch to one or more interfaces on the same switch.

**Configuring for Local SPAN:** Local SPAN configures using “monitor session” command specifying source and destination on the same switch.

```
Switch1# configure terminal
Switch1(config)# monitor session 1 source interface fastEthernet0/2
Switch1(config)# monitor session 1 destination interface fastEthernet0/24
Switch1(config)#end
```

**Figure 3 - Configure Local Span port on older Cisco 2900/3500XL**

Local SPAN configuration syntax on Cisco IOS release 12.2(33)SXH and beyond as shown below.

```
monitor session 1 type local
source int fa0/2
destination int fa0/24
```

**Figure 4 - Configuring Local SPAN port IOS 12.2(33) and later (not ALL Cisco switches)**

A good link for port mirroring example across different networking vendors....


**Remote SPAN (RSPAN):** An extension of SPAN called remote SPAN or RSPAN. RSPAN allows you to monitor traffic from source ports distributed over multiple switches, which means that you can centralize your network capture devices. RSPAN works by mirroring the traffic from the source ports of an RSPAN session onto a VLAN that is dedicated for the RSPAN session. This VLAN is then trunked to other switches, allowing the RSPAN session traffic to be transported across multiple switches. On the switch that contains the destination port for the session, traffic from the RSPAN session VLAN is simply mirrored out the destination port. Not all switches support remote SPAN.
**Configuring RSPAN:** **Step1:** In order to configure RSPAN you need to have a RSPAN VLAN, those VLANs have special properties and can't be assigned to any access ports. To create a VLAN for RSPAN on Cisco IOS, you must create the VLAN via the config-vlan configuration mode, as opposed to using the older VLAN database configuration mode. During the process of defining VLAN parameters, you must specify that the new VLAN is an RSPAN VLAN by configuring the remote-span VLAN configuration command.

```
Switch1# configure terminal  
Switch1(config)# vlan 200  
Switch1(config-vlan)# remote-span  
Switch1(config-vlan)# end  
Switch1# show vlan remote-span  
Remote SPAN VLANs
-----------------------------------------------------------------------------------  
  200  
Switch2# configure terminal  
Switch2(config)# vlan 200  
Switch2(config-vlan)# remote-span  
Switch2(config-vlan)# end  
Switch2# show vlan remote-span  
Remote SPAN VLANs
-----------------------------------------------------------------------------------  
  200  
```

**Figure 5 - Configuring the RSPAN on 'Local' & 'Remote' Cisco switches**

**Step2:** Next configure the RSPAN on Source switch: Unlike SPAN, where the source and destination ports exist on the same switch, the source and destination ports for an RSPAN session reside on different switches. This requires a separate RSPAN source session to be configured, as well as a separate RSPAN destination session to be configured.

```
Switch1# configure terminal  
Switch1(config)# monitor session 1 source interface fastEthernet0/2 rx  
Switch1(config)# monitor session 1 destination remote vlan 200 reflector-port fastEthernet0/24  
Switch1# show monitor  
Session 1  
--------  
Type : Remote Source Session  
Source Ports :  
 Rx : Fa0/2  
Reflector Port : Fa0/24  
Dest RSPAN VLAN : 200  
```

**Figure 6 - Configuring the RSPAN monitor session on the ‘Remote’ switch**
Step 3: Configure the RSPAN on destination switch:

```plaintext
Switch2# configure terminal
Switch2(config)# monitor session 1 source remote vlan 200
Switch2(config)# monitor session 1 destination interface fastEthernet0/3
Switch2(config)# exit
```

Figure 7 - Configuring the RSPAN monitor session on the 'Local' switch

**Note:** The RSPAN VLAN should be allowed in ALL trunks between the involved switches (Source and Destination switches in this case); if you have enabled "pruning" in your network, remove the RSPAN VLAN from the pruning, with the command: `switchport trunk pruning vlan remove <RSPAN VLAN ID>` under the interface configure as trunk.

Encapsulated remote SPAN (ERSPAN): Encapsulated Remote SPAN (ERSPAN), as the name says, brings generic routing encapsulation (GRE) for all captured traffic and allows it to be extended across Layer 3 domains, i.e. cross a WAN.

Aruba Switch SPAN Configuration:

```plaintext
Enable vlan’s used

interface-profile switching-profile "vlan6"
  access-vlan 6
!

Configure a mirroring profile, which will be the destination port where cppm is connected.

interface-profile mirroring-profile "dhcp-span-port-4-vineeth"
  destination gigabitethernet "0/0/5"
!

Configure source port where DHCP server is connected.

interface gigabitethernet "0/0/6"
  mirroring-in-profile "dhcp-span-port-4-vineeth"
  mirroring-out-profile "dhcp-span-port-4-vineeth"
  switching-profile "vlan6"
```

Figure 8 - Configure SPAN on MAS

CPPM Log to debug:

- Enable log level to DEBUG for Async-Netd service in CPPM.
ClearPass Onboard

ClearPass Onboard collects rich and authentic device information from all devices during the onboarding process. Onboard then posts this information to Profile via the Profile API. Since the information collected is definitive, Profile directly classifies these devices into their Category, Family and Name, without having to rely on any other fingerprinting information.

HTTP User-Agent

In some cases, DHCP fingerprints alone cannot fully classify a device. A common example is the Apple family of smart devices; DHCP fingerprints cannot distinguish between an Apple iPad and an iPhone. In these scenarios, User-Agent strings sent by browsers in the HTTP protocol are useful to further refine classification results.

User-Agent strings are collected from:

- ClearPass Guest
- ClearPass Onboard
- Aruba controller through IF-MAP interface

MAC OUI

Mac OUI can be useful in some cases to better classify endpoints. An example is Android devices, where DHCP fingerprints can only classify a device as a generic Android device, but it cannot provide more detail about vendor. Combining this information with MAC OUI, Profile can classify a device as HTC Android, Samsung Android, Motorola Android, etc. MAC OUI is also useful to profile devices such as printers which may be configured with static IP addresses.

ActiveSync plugin support for Exchange 2010 & 2013

ActiveSync plugin is a Windows Service component (that is, it runs as a service on the Exchange server) provided by Aruba to be installed on Microsoft Exchange servers. When a device communicates with the corporate Exchange Server using the ActiveSync protocol, it provides attributes such as device type and user agent. These attributes are collected by the plugin software and are sent to CPPM Profile. Profile uses dictionaries to derive profiles from these attributes.

**Configuration of .NET for ActiveSync plugin 2013**

Whilst we support both Exchange 2010 and 2013 there are a couple of minor nuances to support the plugin on 2013. These are documented below. Whilst trying to install ActiveSync Plugin in Exchange 2013 we may see an error as in the below diagram:

![Figure 9 - Error message to install .NET on Exchange 2013](image)

The issue occurs as the plugin needs the MSFT .NET framework 2.0 to be present in the Exchange server 2013. But Exchange 2013, ships with the 4.5 .NET framework.

In order to overcome this issue, we need to install .NET framework 2.0 on Exchange 2013, **before** installing the Aruba ActiveSync Plugin.

Below are the steps required to install .NET 2.0 on the Exchange server.

1. Navigate to Server Manager - -> Add Roles and Features.
2. Navigate to Features and select .NET Framework 3.5 Features, as shown in the below Screenshot. (.NET 3.5 includes .NET 2.0).
3. While trying to install .NET 3.5 you are required to use the “Specify source ...” option in the summary of the Add/Remove Page and point to the directory [DVD]\Source\SXS (or local copy), because the DLLs .NET 3.5/2.0 are not copied for the winsxs directory where Windows keeps the DLLs in general.

4. After installation you are required to enable the ASP.NET 2.0. Open the wizard again and enable ASP.NET 3.5 in IIS.
Configuration of ActiveSync plugin 2010 and 2013

Once the prerequisite are completed (2013) the general installation can then be followed.

1. The plugin is packaged as ArubaMSExchangePlugin.zip. This contains two files:
   a. setup.exe
   b. MSExchangePlugin.msi
2. Extract and copy both files on Microsoft Exchange Server
3. Double click on "setup.exe" and install the Aruba MSExchange Plugin

Installation Folders

The plugin gets installed under "C:\Program Files\ArubaNetworks\" on 32-bit systems, and under "C:\Program Files (x86)\ArubaNetworks\" on 64-bit systems.

Folder structure is:

- $install_root\bin ==> Contains binaries of MSExchange Plugin
- $install_root\etc ==> Contains configuration files
- C:\ArubaNetworks\MSExchangePlugin\data ==> Contains ActiveSync plugin records which are periodically collected by the plugin
- C:\ArubaNetworks\MSExchangePlugin\var ==> Contains plugin log files

Configuration Files

1. IIS log reader configuration file
Location : $install_root\etc\iislogreader.conf

The contents of the configuration file are pasted below:

```
[iis-log-config]
logDir=C:/inetpub/logs/LogFiles/W3SVC1
####################################################
# If advanced logging is enabled then make sure you
# specify the path for advanced logging files
# in the logDir variable
####################################################
advancedLogging=0
####################################################
# Read interval in seconds
# ReadInterval=300
####################################################
refreshInterval=14400
```

Aruba Networks
2. ActiveSync log record configuration file Location: $install_root\etc\logrecord.conf

Contents of the configuration file are pasted below: Note the section highlighted in RED below which refers to the CPPM node where the plugin transmits data to. The username must be a LOCAL ADMIN-USER (Administration-> Users and Privileges->Admin Users) user configured on the CPPM node with a role of API Administrator.

```plaintext
[log-record-config]
# This is the data directory where the ActiveSync records are stored prior to sending it to Profile
dataDir=C:/ArubaNetworks/MSExchangePlugin/var/data

[log-dispatcher-config]
# This is the Profile URL and login credentials
url=http://<profile-ipaddress>/async_netd/deviceprofiler/endpoints
username=<XXXXXXXXX>
password=<YYYYYYYYY>
```

3. MSExchange Plugin configuration file Location: $install_root\etc\msexchange-plugin.conf

Contents of the configuration file are pasted below:

```plaintext
[domain-controller-info]
# AD domain controller name
serverName=WIN2008R2DEV-AD.dev.avendasys.com
# AD domain controller base dn
baseDn=dc=dev,dc=avendasys,dc=com
# AD domain authentication source name
authSourceName=
# AD domain bind dn
```
Any configuration file changes above require the restart of Aruba MSExchange Plugin service.

The below provides an insight into the data attributes we obtain from the Plugin.

Figure 12 - Activesync Plugin attributes in Endpoint
**CPPM OnGuard**

ClearPass OnGuard agents perform advanced endpoint posture assessment. It collects and sends OS details from endpoints during authentication. Profile uses os_type attribute from OnGuard to derive a profile. For example, a Device Name of Windows XP can be further classified as Windows XP Service Pack 3.

**SNMP**

Endpoint information obtained by reading SNMP MIBs of network devices is used to discover and profile static IP devices in the network. The following information read via SNMP is used:

- **sysDescr** information from RFC1213 MIB is used to profile the device. This is used both for profiling switches/controllers/routers configured in CPPM, and for profiling printers and other static IP devices discovered through SNMP or subnet scans.
- **cdpCacheTable** information read from CDP (Cisco Discovery Protocol) capable devices is used to discover neighbour devices connected to switch/controller configured in CPPM
- **lldpRemTable** information read from LLDP (Link Layer Discovery Protocol) capable devices is used to discover and profile neighbour devices connected to switch/controller configured in CPPM

**Note:** The SNMP based mechanism is only capable of profiling devices if they respond to SNMP, or if the device advertises its capability via Link Layer Discovery Protocol (LLDP).

Prior to CPPM 6.5 when performing SNMP reads for a device, CPPM uses SNMP Read credentials configured in Network Devices, would default to using SNMP v2c with the “public” community string.

Starting in CPPM 6.5, we enhanced the ability to allow multiple SNMP community strings to be defined and used to query static IP devices discovered by the profiler.

In addition the ability to define SNMP community strings that support the following Versions.......
Figure 13 - Setting SNMP community attributes
Discovering endpoint with static IP address

There are two ways to discover endpoints that are statically addressed:

- ARP read
- Subnet scan

Discovery via ARP Read

ARP table read from NAS is used as a means to discover endpoints in the network.

Network Devices configured with SNMP Read enabled are polled periodically for updates based on the time interval configured in Administration -> Server Configuration -> Service Parameters -> ClearPass network services -> Device Info Poll Interval

![Figure 14 - Setting ARP read frequency](image)

The following additional settings have been introduced for the ARP table read:

1. **Read ARP Table Info** – Enable this setting if this is a L3 device and you want to use the ARP table on this device as a way to discover endpoints in the network. Static IP endpoints discovered this way are further probed via SNMP to profile the device.

2. **Force Read** – Enable this to ensure all CPPM nodes in the cluster read SNMP information from this device irrespective of trap configuration on the device. This option is especially useful when demonstrating static IP based device profiling, since this does not require any trap configuration on the network device.

3. In large or geographically spread cluster deployments you do not want all CPPM nodes to probe all SNMP configured devices. The default behavior is for a CPPM node in the cluster to read network device information only for devices configured to send traps to that CPPM node.
Discovery via Subnet Scanner

Network subnet scan is used to discover IP addresses of devices in the network. We use NMAP to discover the devices and whether they have SNMP port 161 open, we then fingerprint these devices to gather additional data. The devices are probed based upon SNMP community strings configured for a SUBNET or HOST address under Configuration-> Networks->Devices. Configuring a device here with an IP or Subnet Address providesProfiler with the SNMP community strings it needs to gather more data. Profiler will use the most specific entry from the Devices list for its SNMP community strings, i.e. if a device is configured with a 172.16.1.0/24 and a SNMP RO string of arubaro but a device in this subnet is configured with an address of 172.16.1.250/32 with a SNMP RO string of danny then for this single device this string is used, for the rest of the subnet arubaro will be used.

**Note:** If no match if found then we will probe devices using the default community string public and type V2c.

When defining the device, the option to select ‘Force Read’ and ‘Read ARP Table Info’ is allowed. This ONLY applies to devices configured with a HOST IP address, not a SUBNET.

Note that if a cluster of CPPM nodes exists, the ‘Force Read’ option results in all nodes in the cluster probing the ARP table of the device which is not desired. If the ‘Force Read’ option is not enabled, device ARP table is read only by the CPPM nodes that are configured as SNMP trap targets in the network device (for Cold Start/Warm Start/Link traps).

![Figure 15 - Setting community string and enabling ARP-read](image)

Subnets to scan are configured per CPPM Zone. This is particularly useful in deployments that are geographically distributed. In such deployments, it is recommended that you assign the CPPM nodes in a cluster to multiple “Zones”, based on the geographical area served by that node, and enable Profile on at least one node per zone. Below we have created an additional zone ‘California’ to that of ‘default’ and then assigned the IP Subnets specific to that new zone as can be seen below.
The frequency of the SUBNET scan is controlled from cluster-wide settings and by default this occurs ONCE every 24-hours. However, in CPPM 6.5 we added the ability to perform “one-time” subnet scans. See below for more detail.

One-Time Subnet Scans

Adding to our ability to scan based upon the Cluster-Wide settings, we’ve added an option for the Administrator to define a SUBNET and perform an immediate subnet scan to profile devices and endpoints. NMAP scans the subnet looking for IP addresses with port 161(SNMP) open. Then a separate scan is triggered for those endpoints discovered to probe for additional context about the endpoint. Profiler post updates to the Event Viewer.
IF-MAP

If configured, Aruba Controller (AOS 6.3 and higher) can send HTTP user-agent and DHCP packets through IF-MAP interface. IF-MAP info sent by a wireless client has mac, ip and user-agent. But wired clients can only provide ip and user-agent, hence dhcp relay has to be properly configured to populate IP-MAC table to fetch the mac address for given IP.

Configurations of IF-MAP on AOS Controller:
To enable IF-MAP on Aruba controller:

In the GUI: follow these steps

Go to: Configuration -> Advanced Services > All Profile Management > Other Profiles-> CPPM IF-MAP
Click Enable: **CPPM IF-MAP Interface** and ADD CPPM details, this will add the CPPM node.

- Host: <CPPM IP Address> or <FQDN>
- Port: 443
- Username: apiadmin
- Password: apiadmin <password>

Configure the username with an admin user who has limited privilege level, API Administrator or Read only access works fine.

![CPPM IF-MAP Interface](image)

**Figure 20 - Enabling Aruba Ctrl to send IF-MAP info to CPPM (GUI)**

**CLI:**

```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>Enables the IFMAP protocol.</td>
<td>=</td>
</tr>
<tr>
<td>server</td>
<td>Configures the CPPM IF-MAP server.</td>
<td>=</td>
</tr>
<tr>
<td>host &lt;host&gt;</td>
<td>IP address/hostname of the CPPM IF-MAP server.</td>
<td>=</td>
</tr>
<tr>
<td>port &lt;port&gt;</td>
<td>Port number for the CPPM IF-MAP server. The range is 1-65535.</td>
<td>443</td>
</tr>
<tr>
<td>username</td>
<td>Username for the user who performs actions on the CPPM IF-MAP server.</td>
<td>=</td>
</tr>
<tr>
<td>passwd &lt;password&gt;</td>
<td>Password of the user who performs actions on the CPPM IF-MAP server. The password must be between 6-100 bytes in length.</td>
<td>=</td>
</tr>
</tbody>
</table>
```

**Example**

This example configures IFMAP and enables it.

```
(host) (config) #ifmap
(host) (config) #ifmap cppm
(host) (CPPM IF-MAP Profile) #server host <host>
(host) (CPPM IF-MAP Profile) #port <port>
(host) (CPPM IF-MAP Profile) passwd <password>
(host) (CPPM IF-MAP Profile) enable
```

**Figure 21 - Enabling Aruba Ctrl to send IF-MAP info to CPPM (CLI)**

**CPPM Logs to debug :**

- Enable log level to DEBUG for IF-MAP from CPASS-Network-Service.
- Enable log level to DEBUG for async-netd service.
Cisco Device Sensor

Device Sensor feature is used to gather raw endpoint data from network devices using protocols such as Cisco Discovery Protocol (CDP), Link Layer Discovery Protocol (LLDP), DHCP and HTTP User-Agent info. All these attributes are sent to CPPM using radius accounting packets. On receiving accounting data, the radius server will post these inputs to profiler for profiling. This feature targets the information gleaned from accounting packets received in CPPM to the profiler component so that endpoints can be profiled without needing IP helper configuration or port SPAN.

Note: Currently this works only with Cisco devices, as specific IOS s/w is required.

Tested Versions

Cisco switch supports [Version 15.0(2)SE2] : DHCP, CDP and LLDP
Cisco controller supports [Version 7.5.102.0] : DHCP and HTTP_User_Agent

Basic Configuration needed:

1. CPPM should be configured with interim accounting packets update enabled.
2. Accounting configuration on NAD.
3. Enable IOS sensor on NAD.

Cisco switch configuration.

1. Basic radius configuration with accounting enabled.
2. Add device-sensor configuration as follows.

Configuration to enable global device sensor in Cisco switch:

```
device-sensor accounting
device-sensor notify all-changes
```

Figure 22 - Enabling device sensor on Cisco switch

Device sensor filter configuration to add what DHCP info in accounting packets.

```
device-sensor filter-list dhcp list dhcp-list
option name host-name [Supported Value 1 : dhcp option 12]
option name parameter-request-list [Supported Value 2 : dhcp option 55]
option name class-identifier [Supported Value 3 : dhcp option 60]
```

Figure 23 - Configuring device sensor on Cisco switch
Device sensor filter configuration to set what LLDP TLV info is in accounting packets.

```plaintext
device-sensor filter-list lldp list lldp-list
tlv name system-description [Supported value 1 : TLV 0006 - lldp_sys_description]
```

Figure 24 - Enabling device sensor LLDP TLV attributes

Device sensor filter configuration to set what CDP info is in accounting packets.

```plaintext
device-sensor filter-list cdp list cdp-list
tlv name version-type [Supported Value 1:TLV0005- cdp_sys_description]
tlv name platform-type [Supported Value 2:TLV0006 - cdp_cache_platform]
```

Figure 25 - Enabling device sensor CDP TLV attributes

Configurations to enable DHCP, LLDP and CDP filter in accounting packets

```plaintext
device-sensor filter-spec dhcp include list dhcp-list
device-sensor filter-spec lldp include list lldp-list
device-sensor filter-spec cdp include list cdp-list
```

Figure 26 - Enabling device sensor filter for DHCP, LLDP & CDP

Globally enable LLDP.

```plaintext
Switch# configure terminal
Switch(config)# lldp run
Switch(config)# end
```

Figure 27 – Globally enable LLDP

Enable LLDP on an interface.

```plaintext
Switch# configure terminal
Switch(config)# interface gigabitethernet1/0/1
Switch(config-if)# lldp transmit
Switch(config-if)# lldp receive
Switch(config-if)# end
```

Figure 28 – Enable LLDP on an interface

Globally enable CDP.

```plaintext
Switch# configure terminal
Switch(config)# cdp run
Switch(config)# end
```

Figure 29 – Globally enable CDP
Cisco WLC Configuration.

1. Login to WLC
2. Configure a WLAN with DHCP profiling.
   1. Go to WLAN configuration Advanced tab.
   2. Enable DHCP Addr. Assignment Required
   3. Enable DHCP profiling and HTTP profiling under option Radius client profiling.

CPPM Log to debug:

- Enable log level to DEBUG for Radius Server.
- Enable log level to DEBUG for Async-Netd service.

Enterprise Mobility Management (EMM/MDM)

Introduction: With the release of ClearPass Policy Manager 6.0.2 and the subsequent release of ClearPass Policy Manager, integration options are now available with the major Enterprise Mobility Management (EMM) platforms, allowing Aruba ClearPass customers to extend the knowledge of managed device state (device type, policy compliance) down to the business rules that govern their corporate network admission policies.

For example, if the EMM platform detects that a device is jailbroken, the EMM platform only has the option to attempt to enforce the business policy at the device level. By extending this policy state to ClearPass as the network policy definition point, the jailbreak status of a device can be used to deny access or quarantine this device the next time it attempts to connect to the secure network.

TechNote: Please review the ClearPass EMM/MDM TechNote for more indepth information about our CPPM and EMM Integration, click here to access this document folder on the support site.

How it works: A service running in CPPM periodically polls EMM servers using their exposed APIs. Device attributes obtained from EMM are added as endpoint tags. Profiler related attributes are send to profiler which uses these attributes to derive final profile

Below we show an example of the additional attributes that can be integrated into the ClearPass Endpoint profiler database that could be received from an EMM vendor. Not all EMM vendors expose the same level of data, but we normalize the information received and present it in a standard attribute template in the endpoint database.
Figure 30 - Example of EMM attributes #1...

Figure 31 - Example of EMM attributes #2...

Figure 32 - Example of EMM attributes #3...
EMM Configuration Details

From the Administration menu of ClearPass Policy Manager, the menu option called Endpoint Context Servers is used to add and configure the EMM Servers.

Use “Add” option to add a specific type of EMM Server, the following figure shows various EMM Servers that are supported by CPPM.

![Endpoint Context Server Details](image)

Figure 33 - Adding an MDM context server

Some minor differences exist in various types of EMM vendors with respect to some parameters for polling and fetching the details. Some of them are shown below, more are detailed in the EMM TechNote.

- Airwatch makes use of an API Key
- MaaS360 makes use of an Application Access Key, Application ID, Application Version, Platform ID and a Billing ID
- SOTI makes use of a Group ID

The polling interval for EMM Servers is configured at the cluster level from Administration > Server Manager > Server Configuration and click on Cluster-Wide Parameters

![Server Configuration](image)

Figure 34 - Setting Cluster Wide Parameters
TCP Fingerprinting

Starting in CPPM 6.5 we added an additional source of Profile context, TCP Fingerprinting. To enable this feature you must enable not be using the Data Port (it must not have an IP address) on a 500 or 5K appliance. On a 25K appliance you can utilize one of the other spare interface, thus the Data Port can be utilized. Within a VM environment if the DATA Port is being used then the ability to use TCP Fingerprinting is not an option.

Following the configuration of the SPAN port, ensure that the switch port is actually ‘spanning’ data from the network. CPPM will then analysis the SYN, SYN-ACK handshakes utilizing industry recognized databases, pf0.fp = SYN DB and pf0fa.fp = SYN & ACK DB. This allows CPPM to work out who (the client) is connecting to a server (the SYN), then looking at the SYN-ACK allows CPPM to derive what the actual server (target) is.

This is passive analysis of the data flows on the network, but please do remember that TCP Fingerprinting is a resource intensive process and enabling this within the CPPM network needs careful consideration. If your in doubt consult with a specialist before enabling this.

Figure 37 - Enabling TCP Fingerprinting 'Warning'

An example of a TCP Fingerprint, this is used to identify the host in more details.

Figure 38 - Example of a TCP Fingerprint
Profiling

Profile uses a **two-stage** approach to classifying endpoints using input attributes.

**Stage 1**

Stage 1 tries to derive device profiles using static dictionary lookups. Based on the attributes available, CPPM looks up DHCP, HTTP, ActiveSync and MAC OUI dictionaries, and derives multiple matching profiles. Each attribute from a source (eg DHCP, SNMP etc) is assigned 2 weights – **reliability** and a **score**.

If profiling results in multiple matches, these weights are used to find best match. All matches are sorted on (reliability, score) tuple and one with highest value is chosen.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Reliability</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>dhcp:options</td>
<td>98</td>
<td>95</td>
</tr>
<tr>
<td>dhcp:option55</td>
<td>98</td>
<td>95</td>
</tr>
<tr>
<td>dhcp:option60</td>
<td>99</td>
<td>96</td>
</tr>
<tr>
<td>snmp:sys_descr</td>
<td>100</td>
<td>97</td>
</tr>
<tr>
<td>snmp:cdp_cache_platform</td>
<td>100</td>
<td>97</td>
</tr>
<tr>
<td>snmp:device_type</td>
<td>98</td>
<td>1</td>
</tr>
<tr>
<td>snmp:name</td>
<td>98</td>
<td>2</td>
</tr>
<tr>
<td>host:os_type</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>host:user_agent</td>
<td>10</td>
<td>99</td>
</tr>
<tr>
<td>active_sync:device_type</td>
<td>100</td>
<td>99</td>
</tr>
<tr>
<td>active_sync:user_agent</td>
<td>100</td>
<td>99</td>
</tr>
</tbody>
</table>

**Figure 39 - Profiling Reliability/Score**

In addition to these attributes, mac_vendor and hostname are also used in Stage-2 rule evaluation.
**Example:** In this example an Aruba controller proxied HTTP requests from an Apple iPad. HTTP User-Agent classifies the device as an Apple iPad. SNMP collector provides sys_descr which classifies the device as Aruba controller. As the **device_category** of profile derived from these 2 inputs are different (Computer, Controller), CPPM's profiles picks the one with highest reliability and finally classifies this device as Aruba Controller.

```bash
curl -X POST http://localhost:6180/async_netd/deviceprofiler/endpoints \
-H "Content-Type: application/json" -d \n'"mac" : "000b86625750",
"host": {
  "user_agent": "iPad;",
},
"snmp": {
  "sys_descr": "ArubaOS (MODEL: Aruba620), Version"
}
```

**Stage 2**

CPPM comes pre-built with a set of rules that evaluates a device profile. CPPM uses all input attributes and device profiles from **Stage 1**. The resulting rule evaluation may or may not result in a profile. **Stage 2** is intended to refine the results of profiling.

**Example:** DHCP option55 classifies device as Android. Stage 2 rules reclassifies the device as HTC Android by combining mac-vendor information.

```bash
curl -X POST http://localhost:6180/async_netd/deviceprofiler/endpoints \
-H "Content-Type: application/json" -d \n'
"mac" : "00092d112233",
"hostname": "myandroid.domain.com",
"dhcp": {
  "options": ["53,55,57,61,51"],
  "option55": ["1,121,33,3,6,12,15,28,51,58,59,119", "]
}
}';
```
**Post Profile Actions**

After profiling an endpoint, profile can be configured to perform RADIUS Change of Authorization (CoA) on the NAD to which an endpoint is connected. Post profile rules are configured in the CPPM Service configuration wizard. Make sure you turn on “Profile Endpoints” from the Service tab:

![Figure 40 - Enabling Profiler on a service](image)

Configure [Endpoints Repository] as Authorization Source. Endpoint profile attributes derived by Profile are available through the ‘[Endpoints Repository]’ authorization source. These attributes can be used in role-mapping or enforcement policies to control network access. Available attributes are:

- **Authorization:[Endpoints Repository]:MAC Vendor**
- **Authorization:[Endpoints Repository]:Category**
- **Authorization:[Endpoints Repository]:OS Family**
- **Authorization:[Endpoints Repository]:Name**

![Figure 41 – Using [Endpoints Repository] as Authorization Source](image)
You can select a set of categories and a CoA profile to be applied when the profile matches one of the selected categories. CoA is triggered using the selected CoA profile. ANY option from ‘**Endpoint Classification**’ can be used to invoke CoA on a change of any one of the fields (category, family, and name).

![Config](image)

**Figure 42 - Send CoA based upon endpoint classification**

Use profiled endpoint attributes in Role Mapping Rules.

![Config](image)

**Figure 43 - Example of using Profiled info in role-mapping**
Fingerprint Dictionaries

CPPM uses a set of dictionaries and built-in rules to perform device fingerprinting. Listed below are the dictionaries used by CPPM.

- DHCP
- HTTP User-Agent
- ActiveSync attributes
- SNMP attributes
- MAC OUI

![Device Fingerprint Dictionaries in CPPM](image)

**Figure 44 - Example set of Device Fingerprint Dictionaries in CPPM**

As these dictionaries can change frequently, CPPM provides a way to automatically update fingerprints from an Aruba hosted portal. If external access cannot be provided to CPPM, the fingerprints file can be downloaded and imported through CPPM admin. The following screenshots show the configuration details for online and manual fingerprint updates.
Figure 45 - CPPM WEB s/w Update

Profile Redundancy

If profiling is enabled on multiple nodes within a zone, they will form a cluster which provides redundancy and load balancing. The node with lowest UUID assumes an active role. All other nodes proxy endpoint attributes to active profiler. Active profiler periodically sends heartbeats to peers. If active node goes down, heartbeats will be lost and next peer with lowest UUID assumes master role.

When failed node comes back, it will start sending heartbeats and assumes master role. If any peer has assumed master role, it will change to passive role on receiving heartbeats from original master.

Profiler Load Balancing

Collectors can run on any node and can proxy extracted attributes to active profiler. This property of profiler helps to spread load across multiple CPPM nodes.

Example: DHCP relay or span is configured to a CPPM node which is not enabled as profiler. This node can perform required packet processing, extract mac, ip, hostname, option55, option60 and send to active profiler.
Profile UI

CPPM provides user interfaces to search and view profiled endpoints. It also provides basic statistics on the profiled endpoints.

*Dashboard widget showing basic distribution of device types*

![Dashboard widget showing basic distribution of device types](image)

*Figure 46 - Dashboard Widgets for profiling*
**Detailed device distribution and list of endpoints**

**Figure 47 – Summary of Profiler Endpoint Information**
Profile details of an endpoint

![Image of endpoint profile details]

Figure 48 - Detailed Profiler endpoint information

Search endpoint profiles based on category/family/name, etc.

![Image of endpoint search]

Figure 49 - Complex search of endpoint DB based upon Profiler attributes
Profile APIs

Profile exposes a set of REST APIs to receive endpoint attributes and to provide results of profiling. Basic HTTP authentication using CPPM admin user/passwords are required for the APIs. Third-party products can easily integrate with ClearPass Profile by writing to these APIs.

Post endpoint attributes for profiling

Attributes for a single or multiple endpoints can be POSTed to the following URL; this triggers profiling. MAC or IP address has to be present as the key. Other attributes are optional. If IP address is used as the key, Profile should have received MAC-IP binding from other sources such as DHCP. If `device:{category, family, name}` is posted, profiler will ignore other inputs and considers this as authentic profile.

- URL: https://{host}/async_netd/deviceprofiler/endpoints
- Method: POST
- Content-Type: application/json
- Input: Single or list of endpoint attributes

```json
{
    mac:
    ip:
    dhcp : {
        option55:
        option60:
        options:
    }
    hostname:
    active_sync : {
        device_type:
        user_agent:
    }
    host: {
        os_type:
        user_agent:
    }
    snmp: {
        sysdescr:
        device_type:
        cdp_cache_platform:
    }
    device: {
        category:
        family:
    }
}
```
Get endpoint by MAC or IP address

- **URL:** [https://device-profiler/async_netd/deviceprofiler/endpoints/{mac/ip}](https://device-profiler/async_netd/deviceprofiler/endpoints/{mac/ip})
- **Method:** GET
- **Output:**
  - 200 OK - Success with json encoded endpoint details

```json
{
  ip:          => endpoint ipaddress
  hostname:    => endpoint hostname
  device_category: , => Computer, SmartDevice, Printer etc
  device_family: , => Android, Apple, Windows etc
  device_name: , => Samsung Android, Apple iPad etc
  added_at:    => as unix timestamp in seconds
  updated_at:  => as unix timestamp in seconds
}
```

- 404 Not Found - if endpoint with given MAC or IP address does not exist.
- 500 Internal Error - on service internal errors
Custom Device/Fingerprint Classification (added in CPPM 6.5)

Building on the previously released Policy Manager API’s, in 6.5 CPPM allows an administrator to perform custom device classification of **unknown** devices. Basically we allow admin’s to create custom rules from an endpoint using profiled attributes like

- hostname
- mac_vendor
- fingerprint details from
  - "dhcp.option55"
  - "dhcp.option60"
  - "snmp.sys_descr"
  - "host.user_agent"
  - "host.os_type"
  - "nmap.device"
  - "tcp.device"
  - "active_sync.device_type"

and then re-profile all other device that have similar pattern.

The Work Flow - Overview

1. If the expected device is not in dictionary, create it using the Device Dictionary API.
2. Manually profile the device using the API, with newly added device info.
3. Invoke Rules API for the above profiled endpoint using its **mac** and **rule_fields**. This will automatically create a new rule in DB.
4. Using the re-profile API, triggers re-profiling of all **unknown endpoints**, which will profile all unknown endpoints that matches the above rule.

Also,

5. Update/Delete bad/accidental devices using combination of Device DELETE & POST API.
6. Update/Delete bad/accidental rules using combination of Rules DELETE & POST API.

**Note:**

In 6.5 custom device profile APIs have to be invoked from the **Publisher** node only.

Following is a break down and examples of the API’s, we’ve used the cURL command as our interface to drive the API’s to test the functionality. You could achieve the similar results using other tools such as wget
**API’s**

**Device Dictionary API:**

1. **API to ADD New Definition to Dictionary:**
   Create a new definition in the dictionary if it is not already present. The API returns an id of the device created. Dictionary created using API will have ids starting from **100000**.

   **Method:** POST
   **URL:** /async_netd/deviceprofiler/devices
   **Values:**
   ```json
   {
   "device_category": "",
   "device_family": "",
   "device_name": ""
   }
   ```

   **Example:**
   **CMD:**
   "device_category": "SmartDevice",
   "device_family": "Future-iPhone",
   "device_name": "iPhone20"
   }';

   **Output:** 100000 [New device ID]

---

**Figure 50 – Checking the definition was created in the SmartDevice category**
2. **API to Query/List Custom Device from Dictionary:**

This API allows admins to query devices from a dictionary. All query params are optional and the query uses a prefix match. If no query params are provided, all devices are returned.

**Query with filter:**

This API allows the query a device for given `device_category`, `device_family`, `device_name`

**Method:** GET  
**URL:**  
/async_netd/deviceprofiler/devices?device_category={}&device_family={}&device_name={}

**Example:**  
**CMD:**  
curl -u apiadmin:password -k https://<CPPM IP>/async_netd/deviceprofiler/rules?device_name=Test

**Output:**

```
[
  {
    "device_family": "Test",
    "id": 100000,
    "device_category": "SmartDevice",
    "device_name": "Test 005"
  }
]
```

**Query all devices:**

This API allows you to query all dynamically added devices.

**Method:** GET  
**URL:** /async_netd/deviceprofiler/devices/x

**CMD:**  
curl -u apiadmin:password -k https://<CPPM IP>/async_netd/deviceprofiler/devices/x

**Output:**

```
[
  {
    "device_family": "Future-iPhone",
    "id": 100000,
    "device_category": "SmartDevice",
    "device_name": "iPhone20"
  },
  {
    "device_family": "TME",
    "id": 100001,
```
"device_category": "SmartDevice",
"device_name": "Dannyj"
]

3. API to Delete Custom Device

This API allows admins to delete custom devices from a dictionary. Admin should initially identify the correct device id which needs to be deleted and pass it through the URL.

**METHOD:** DELETE

**URL:** /async_netd/deviceprofiler/devices?id={}

**Example:**

**CMD:**
curl -X DELETE -u apiadmin:password -k https://<CPPM IP>/async_netd/deviceprofiler/devices?id=100000

**Output:**
<html>
<head>
<title>200 OK</title>
</head>
<body>
<h1>200 OK</h1>
<br /><br />
</body>
</html>

**Note:**
An admin is not allowed to delete devices from the dictionary that is associated with a rule.
RULES API:

1. **API to ADD Rules:**

   Given an unknown endpoint, this API will automatically create rules by **ANDing** rule attributes from endpoint mac_vendor, hostname and fingerprints (ex: "dhcp.option55", "dhcp.option60", "snmp.sys_descr", "host.user_agent", "host.os_type", "nmap.device", "tcp.device", "active_sync.device_type..."). Rules created using API will have ids starting from 100000.

   **Method:** POST  
   **URL:** /async_netd/deviceprofiler/rules  
   **Values:**  
   ```
   {
     mac:
     rule_fields: [...]
   }
   ```
   Where rule_fields = mac_vendor, hostname, dhcp.option55, dhcp.options, dhcp.option60, snmp.sys_descr, host.user_agent etc...

   **Example:**  
   **CMD:**
   ```
curl -X POST -u apiadmin:password https://<CPPM IP>/async_netd/deviceprofiler/rules
   -H "Content-Type: application/json" -k -d
   '{"mac": "6cadf8112341",
    "rule_fields": ["mac_vendor", "dhcp.option55"]
   }'
   ```

   **Output:** 100000 [New rule ID]

2. **API to DELETE Rules:**

   This API allows an admin to delete dynamically created rules. **Note:** Admins should initially identify the correct rule-id that needs to be deleted and pass it through the URL.

   **Method:** DELETE  
   **URL:** /async_netd/deviceprofiler/rules/{id}

   **Example:**  
   **CMD:**
   ```
curl -X DELETE -u apiadmin:password -k https://<CPPM IP>/async_netd/deviceprofiler/rules/100000
   ```
3. API to Query Rules:
This API's allows admins to search previously custom added rules. API's allows you to query rules for given device_name and also query the entire dynamic rules.

Query rules with device_name:
This API allows admin to search rules that could classify fingerprint to a given device.

**Method:** GET  
**URL:** `/async_netd/deviceprofiler/rules?device_name={}`

**Example:**
**CMD:**
curl -u apiadmin:password -k https://<CPPM IP>/async_netd/deviceprofiler/rules?device_name=Test

**Output:**
```json
[
  {
    "id": 100000,
    "rule": {
      "combining_op": "all",
      "conditions": [
        {
          "mac_vendor": "contains",
          "CISCO SYSTEMS, INC."
        },
        {
          "dhcp.option60": "contains",
          "dhcpcd-6.2.10"
        }
      ],
      "device_id": 100000
    }
  }
]
```

Query all rules:
This API allows admin to query all rules dynamically created.

**Method:** GET  
**URL:** `/async_netd/deviceprofiler/rules/x`

**Example:**
**CMD:** curl -u apiadmin:password -k https://<CPPM IP>/async_netd/deviceprofiler/rules/x

**Output:**

```json
[
    {
        "id": 100000,
        "rule": {
            "combining_op": "all",
            "conditions": [
                [
                    "mac_vendor",
                    "contains",
                    "CISCO SYSTEMS, INC."
                ],
                [
                    "dhcp.option60",
                    "contains",
                    "dhcpcd-6.2.10"
                ]
            ]
        },
        "device_id": 100000
    },
    {
        "id": 100001,
        "rule": {
            "combining_op": "all",
            "conditions": [
                [
                    "mac_vendor",
                    "contains",
                    "VMware, Inc."
                ],
                [
                    "tcp.device",
                    "contains",
                    "Linux 2.2.x"
                ]
            ]
        },
        "device_id": 100001
    }
]
```
**Re-profile API:**

API will result in re-profiling all devices with unknown profile. This **could be a costly** operation if there are lots of unknowns.

**Method:** POST  
**URL:** /async_netd/deviceprofiler/endpoints/unknowns

**Example:**
**CMD:**
curl -X POST -u apiadmin:password -k https://<CPPM IP>/async_netd/deviceprofiler/endpoints/unknowns

**Manual Profile API:**

This API allows an admin to manually profile a given endpoint to specific device category, device family and device name. By manually profile, we mean ASSIGN the required classification to the device. This can be anything you want. Below I assign a family of ‘Future-iPhone’ under the SmartDevice category, then I call the Phone an iPhone20 to demonstrate the power of the API's.

**Figure 51 – Checking the device is classified as required**
Method: POST
URL: /async_netd/deviceprofiler/endpoints
Value:
{"mac": "",
"device": {
  "category": "",
  "family": "",
  "name": ""
}
}

Example:
CMD:
danny-jump:~ djump$ curl -X POST
https://10.2.100.161/async_netd/deviceprofiler/endpoints -u apiadmin:arubans123 -H
"Content-Type: application/json" -k -d '{"mac": "00000000013", "device": {"category": "SmartDevice", "family": "Future-iPhone", "name": "iPhone20"}}'
<html>
<head>
<title>200 OK</title>
</head>
<body>
<h1>200 OK</h1>
<br />
</body>
</html>danny-jump:~ djump$