

Automated scanning of SSID, BSSID, and Signal of available wireless APs

Goal: Create a station and scan for SSID, BSSID, and Signal of available wireless APs

We will learn how to use a script to create a station and scan for available APs. We will then look at the /scanresults/ URI and the info we can get from a scan through JSON. Please refer to `sta_scan_test.py` as an example script.

1.

Using the Script

A. Command Line Options

```
--sta_name nameOfStation
```

Specifies the name of the station to be created, if this option is used, the name will default to **sta0000**.

```
--ssid nameOfNetwork
```

Specifies the name of the network to connect to.

This value must be used, however, the SSID does not have to exist and a fake name can be used.

```
--security {WEP, WPA, WPA2, WPA3, Open}
```

Specifies the security type of the network to connect to.

This value must be used, however, if a fake SSID is used the type should be open.

B. Running the script

A. As an example, we can run the script using:

```
./sta_scan_test.py --sta_name sta0000 --ssid fake_ssid --security open --radio
```

B. This will produce output that looks like this:

```
BSS          Signal  SSID
08:36:c9:e3:d4:da  -32.0  Logan-Test-Net
10:56:11:0c:04:02  -80.0  :)
22:56:11:0c:04:02  -79.0  xfinitywifi
32:56:11:0c:04:02  -80.0  NA
```

This script produces limited output, for more detail we can look at the webpage hosted by LANforge.

2.

The /scanresults/ URI

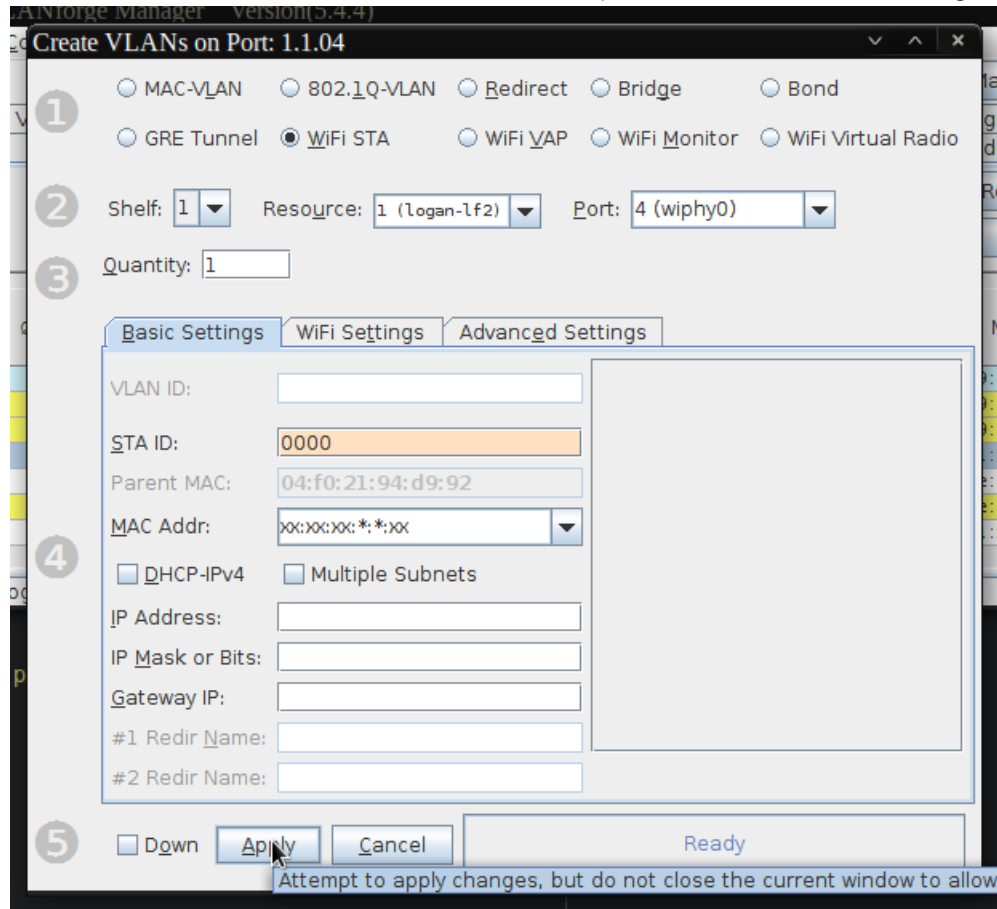
A. In order to view this page we will need to create a station and start a scan.

A. First we will create the station (Make sure to click on a radio in the Port Mgr tab first):

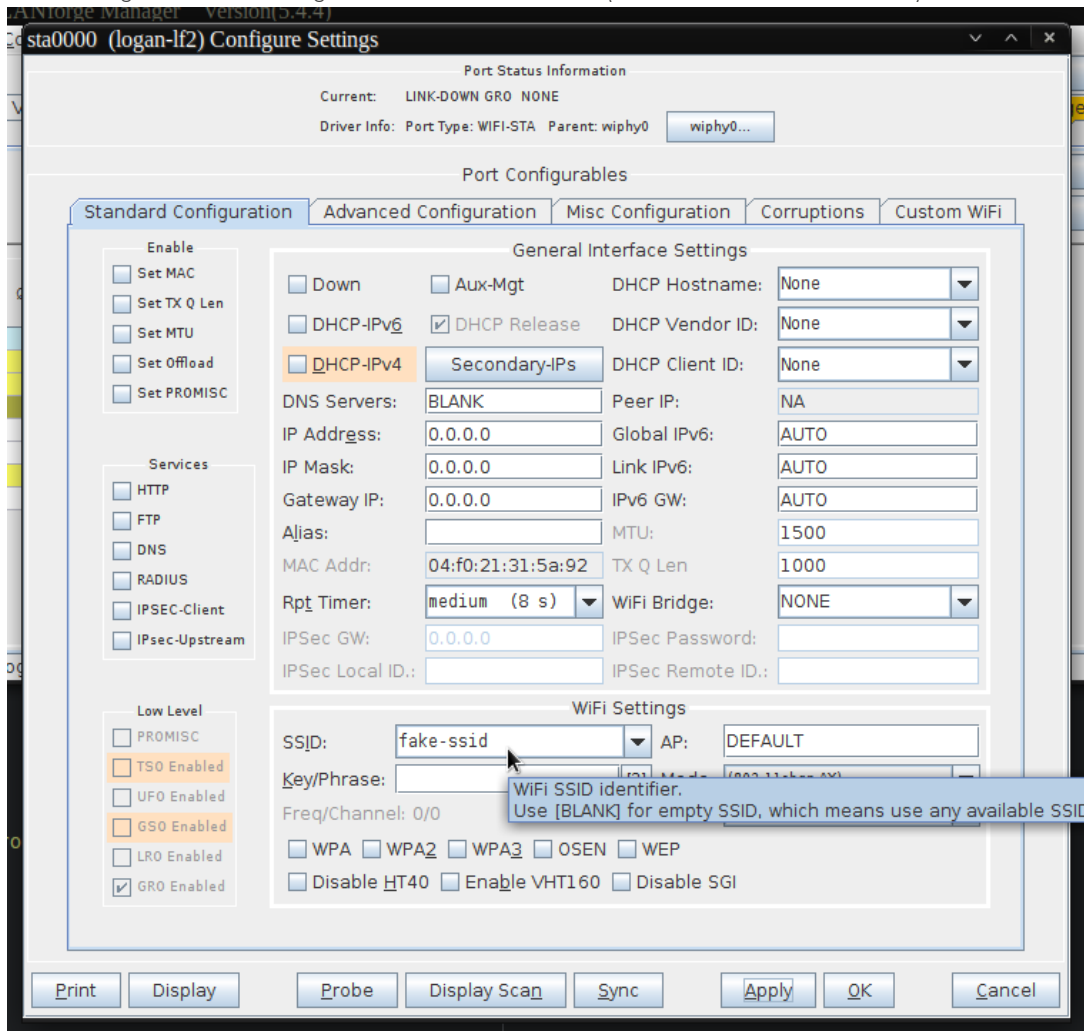
The screenshot shows the LANforge Manager interface. The 'Port Mgr' tab is active, displaying a table of Ethernet interfaces. A tooltip for the 'Create' button reads: 'Create a virtual interface of some type. ALT-E and CTRL-E are the hot-key accelerators'.

ID	IP	Alias	Parent Dev	AP	Channel	SSID	MAC
	192.168.10.20	eth0					00:0d:b9:56:ad:e8
	0.0.0.0	eth1					00:0d:b9:56:ad:e9
	0.0.0.0	eth2					00:0d:b9:56:ad:ea
	0.0.0.0	wiphy0			0		04:f0:21:94:d9:92
	0.0.0.0	wiphy1			0		00:0e:8e:5c:63:82
	0.0.0.0	wlan1	wiphy1		1		00:0e:8e:5c:63:82
	192.168.1.2	wlan0	wiphy0	08:36:C9:E3:D4:DA	44	Logan-Test...	04:f0:21:94:d9:92

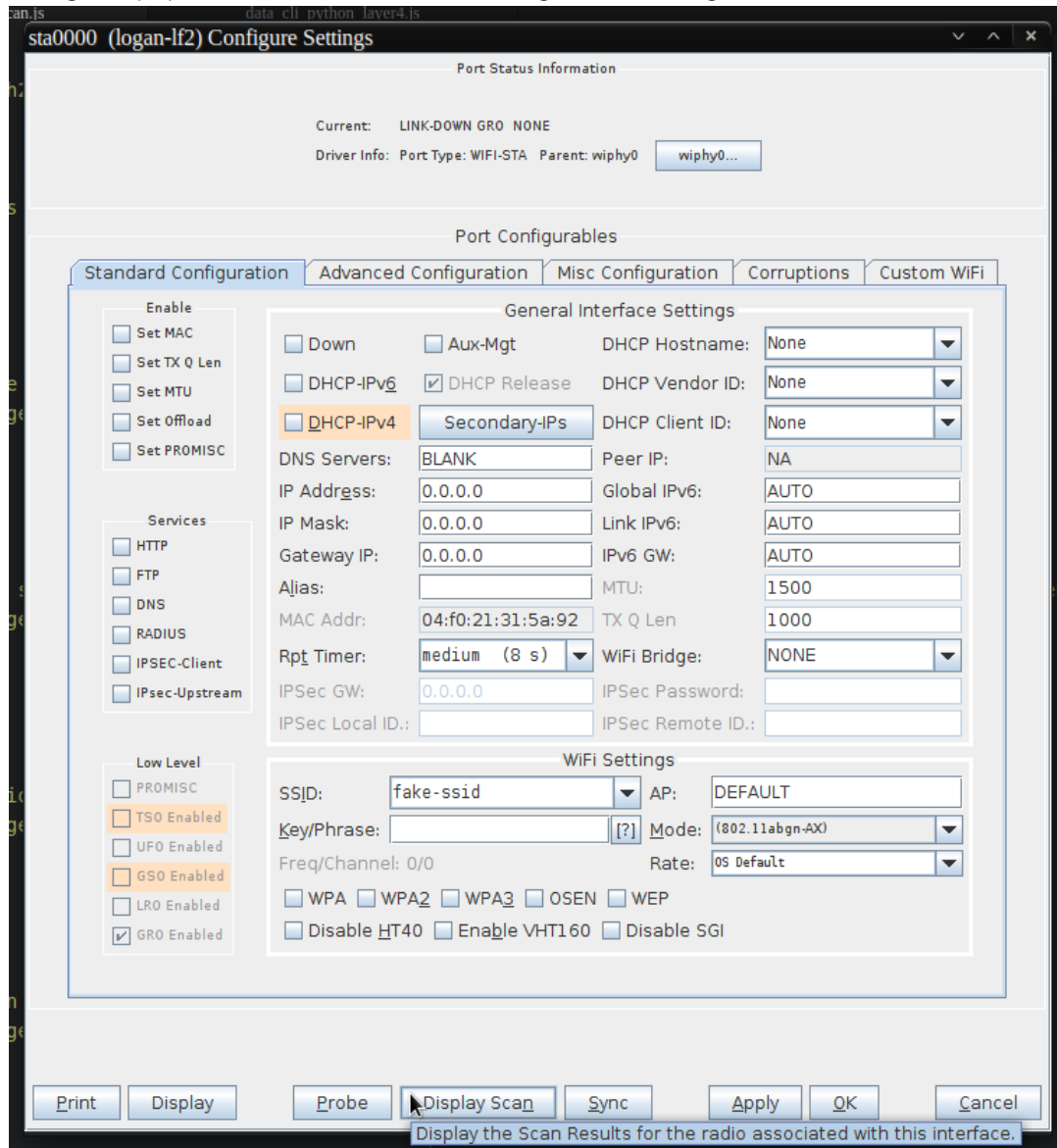
B. Next we will create the station, the default values can be used or a specific number for the station can be given:



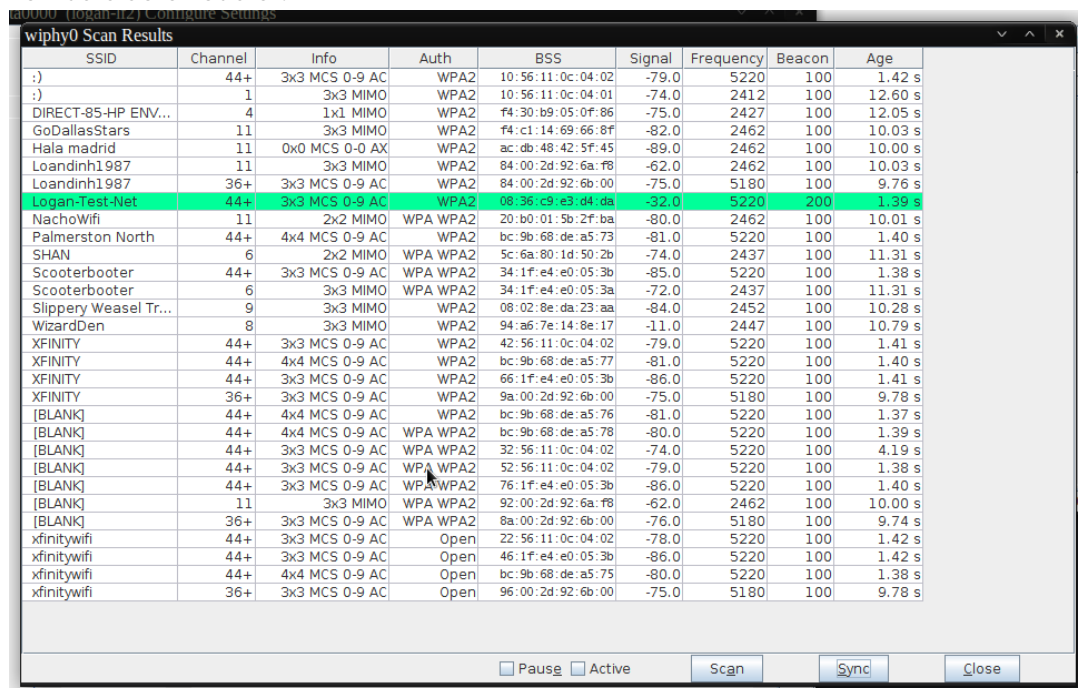
C. After creating the station, we will give the an SSID to connect to. (This doesn't have to be a real AP):



D. Clicking on Display Scan at the bottom of the station settings window will bring us to the Scan window:



E. Finally we'll be able to start the scan and see the results. Clicking on Scan and waiting a few seconds will show all of the APs available to the station:



3.

JSON Response from /scanresults/

- A. Another way of viewing the same information is to use the /scanresults/ URI. This URL can be found at your LANforge ip using port 8080. Ex: 192.168.10.20:8080/scanresults. We will also need the shelf number, the resource number, and the station name. The final URL would look like this
192.168.10.20:8080/scanresults/1/1/sta0000
- B. The scan results can be viewed through JSON by using cURL on the same URL as before. The response will look like this:

```
{"handler": "candela.lanforge.HttpStationScan$FixedJsonResponder", "uri": "scanresults/:shelf_id/:resource_id/:port_id", "candela.lanforge.HttpStationScan": {"duration": "1"}, "scan-results": [{"1.1.4.08:36:c9:e3:d4:da": {"age": "2238", "auth": "WPA2", "beacon": "200", "bss": "08:36:c9:e3:d4:da", "channel": "44", "entity id": "1.1.4", "frequency": "5220", "info": "3x3 MCS 0-9 AC", "signal": "-32.0", "ssid": "Logan-Test-Net"}}]}
```

4.

Accessing and Printing JSON Response with Python

- A. We will use `sta_scan_test.py` as an example for a `start()` method
- A. First, we'll need to send a JSON post using realm. Use [this cookbook](#) as reference for getting started with realm. Our JSON will look something like this:

```
data = {
    "shelf": 1,
    "resource": 1,
    "port": self.sta_list
}
```

- B. We can then use `json_post` to send the request. We'll need to wait about 15 seconds to give the scan time to happen

```
self.json_post("/cli-json/scan_wifi", data)
time.sleep(15)
```

C.

Next, we'll create a variable with the results from the scan using

```
scan_results = self.json_get("scanresults/1/1/%s" % ','.join(self.sta_list))
```

- D. Finally, we'll create a loop to iterate through the JSON response and print some nicely formatted output

```
print("{0:<23}".format("BSS"), "{0:<7}".format("Signal"), "{0:<5}".format("SSID")
for result in scan_results['scan-results']:
    for name, info in result.items():
        print("%s\t%s\t%s" % (info['bss'], info['signal'], info['ssid']))
```

B. Final Results

- A. Our final function will look like this:

```
def start(self):
    self.station_profile.admin_up()
    print(self.sta_list)
    print("Sleeping 15s while waiting for scan")
    data = {
        "shelf": 1,
        "resource": 1,
        "port": self.sta_list
    }
    self.json_post("/cli-json/scan_wifi", data)
    time.sleep(15)
    scan_results = self.json_get("scanresults/1/1/%s" % ','.join(self.sta_list))

    print("{0:<23}".format("BSS"), "{0:<7}".format("Signal"), "{0:<5}".format("SSID")
    for result in scan_results['scan-results']:
        for name, info in result.items():
            print("%s\t%s\t%s" % (info['bss'], info['signal'], info['ssid']))
```

- B. Our formatted output should look like this:

BSS	Signal	SSID
00:0e:8e:52:4e:82	-33.0	test-net
08:36:c9:e3:d4:db	-31.0	Logan-Test-Net
08:36:c9:e3:d4:dc	-27.0	Logan-Test-Net

