sales@candelatech.com
support@candelatech.com
+1 (360) 380-1618 [PST, GMT -8]

LANforge Scripts Users Guide

Script Table of Contents

py-scripts/lf_base_interop_profile.py

This script is a standard library which support different functionality of interop including wi-fi connectivity on all kinds of real clients.

py-scripts/lf_chamberview_tools.py

This script will check if chamberview test is running

py-scripts/lf_cleanup.py

This script is used for cleaning the cross-connections, layer-3-endpoints, stations and bridges in Lanforge. This script is also used to sanitize the lanforge unit, which means will clean the Port Mgr, Layer-3, L3 Endps, and Layer 4-7 tabs.

py-scripts/lf_client_visualization.py

The lf_client_visualization script is used to Monitor the connection status of all the clients for user specified duration. This report shows the connection status of all the clients in the test. This information is very useful when running long duration tests with 1000s of WiFi clients connecting across various bands, channels and SSIDs. The report shows over time counts of number of clients in scanning, connect and IP address acquired states. The report also shows number of clients connected over time per SSID, per Channel, per band and per client type

py-scripts/lf_continuous_throughput_test.py

The Candela WiFi Continuous Rotation test is designed to quickly test throughput and other metrics at different rotations, attenuation, and other configured values. For each combination of settings, it will start traffic and rotate the turntable, gathering stats often to generate report data at different rotations. The traffic test will be stopped only at the end of each rotation.

py-scripts/lf_create_vap_cv.py

Theis script is designed to create a Virtual Access point (VAP) using Chamber view scenario. Through this script, dhcp min, max ranges can be set to the vap and can modify vap ip address, vap ip mask, gateway, vap mode, vap bandwidth, vap frequency, security. Any existing scenario can be loaded or deleted by providing respective arguments.

py-scripts/lf_create_wanlink.py

This script creates a wanlink using the lanforge api.

py-scripts/lf_csv.py

This script is designed to generate the csv file with the test report data of lanforge tests, any file which runs the tests can import this file to generate the csv file with the test report data. The file name is test.csv and the file will the stored in the same directory of this file (pyscripts) while creating the object of this lf_csv file, the data can be passed while creating the object and the report can be generated by calling the method generate_csv() Note this is not the kpi csv format. Use lf_kpi_csv.py for the kpi csv format.

py-scripts/lf_dataplane_test.py

Example report: dataplane.pdf

The Candela Tech WiFi data plane test is designed to conduct an automatic testing of all combinations of station types, MIMO types, Channel Bandwidths, Traffic types, Traffic direction, Frame sizes etc… It will run a quick throughput test at every combination of these test variables and plot all the results in a set of charts to compare performance. The user is allowed to define an intended load as a percentage of the max theoretical PHY rate for every test combination. The expected behavior is that for every test combination the achieved throughput should be at least 70%% of the theoretical max PHY rate under ideal test conditions. This test provides a way to go through hundreds of combinations in a fully automated fashion and very easily find patterns and problem areas which can be further debugged using more specific testing.

py-scripts/lf_ftp.py

lf_ftp.py will verify that N clients are connected on a specified band and can simultaneously download/upload some amount of file data from the FTP server while measuring the time taken by clients to download/upload the file.

py-scripts/lf_graph.py

This script facilitates the generation of comprehensive graphical reports. It offers a variety of graph types, including bar graphs, horizontal bar graphs, scatter graphs, bar-line graphs, stacked graphs, horizontal stacked graphs, and line graphs.

py-scripts/lf_interference.py

Use this script to create co-channel and adjacent channel interference, by changing frequency argument. Provide same channel for co-channel interference, and adjacent channel for adjacent channel interference. This script uses the lf_wifi_capacity_test.py

py-scripts/lf_interop_modify.py

Call commands/modifications to Interop Devices

py-scripts/lf_interop_pdu_automation.py

lf_interop_pdu_automation.py is a stand-alone automation script which will automatically power on/off for certain interval in loop to prevent the mobile phones over charging.

py-scripts/lf_interop_ping_plotter.py

The Candela Tech ping plotter test assesses the network connectivity for specified clients by measuring Round Trip data packet travel time. It also detects issues like packet loss, delays, and response time variations, ensuring effective device communication and identifying connectivity problems.

py-scripts/lf_interop_ping.py

Example report: interop_ping.pdf

The Candela Tech ping test is to evaluate network connectivity and measure the round-trip time taken for data packets to travel from the source to the destination and back. It helps assess the reliability and latency of the network, identifying any packet loss, delays, or variations in response times. The test aims to ensure that devices can communicate effectively over the network and pinpoint potential issues affecting connectivity.

py-scripts/lf_interop_port_reset_test.py

The LANforge interop port reset test enables users to use real Wi-Fi stations and connect them to the Access Point being tested. It then disconnects and reconnects a given number of stations at different time intervals. This test helps evaluate how well the AP handles a dynamic and busy network environment with devices joining and leaving the network at random times. The test will basically disconnect & reconnect to the same network with real devices such as android, linux, windows and generate a report.

py-scripts/lf_interop_qos.py

The Interop QoS test is designed to measure performance of an Access Point while running traffic with different types of services like voice, video, best effort, background. The test allows the user to run layer3 traffic for different ToS in upload, download and bi-direction scenarios between AP and real devices. Throughputs for all the ToS are reported for individual devices along with the overall throughput for each ToS. The expected behavior is for the AP to be able to prioritize the ToS in an order of voice,video,best effort and background. The test will create stations, create CX traffic between upstream port and stations, run traffic and generate a report.

py-scripts/lf_interop_real_browser_test.py

The Candela Web browser test is designed to measure the Access Point performance and stability by browsing multiple websites in real clients like android, Linux, windows, and IOS which are connected to the access point. This test allows the user to choose the options like website link, the number of times the page has to browse, and the Time taken to browse the page. Along with the performance other measurements made are client connection times, Station 4-Way Handshake time, DHCP times, and more. The expected behavior is for the AP to be able to handle several stations (within the limitations of the AP specs) and make sure all clients can browse the page.

py-scripts/lf_interop_rvr_test.py

lf_interop_rvr_test.py will measure the performance of stations over a certain distance of the DUT. Distance is emulated using programmable attenuators and throughput test is run at each distance/RSSI step.

py-scripts/lf_interop_throughput.py

The Client Capacity test and Interopability test is designed to measure an Access Point's client capacity and performance when handling different amounts of Real clients like android, Linux, windows, and IOS. The test allows the user to increase the number of clients in user-defined steps for each test iteration and measure the per client and the overall throughput for this test, we aim to assess the capacity of network to handle high volumes of traffic while each trial. Along with throughput other measurements made are client connection times, Station 4-Way Handshake time, DHCP times, and more. The expected behavior is for the AP to be able to handle several stations (within the limitations of the AP specs) and make sure all Clients get a fair amount of airtime both upstream and downstream. An AP that scales well will not show a significant overall throughput decrease as more Real clients are added.

py-scripts/lf_interop_video_streaming.py

The Candela Web browser test is designed to measure an Access Point's client capacity and performance when handling different amounts of Real clients like android, Linux, windows, and IOS. The test allows the user to increase the number of clients in user-defined steps for each test iteration and measure the per client and the overall throughput for this test, we aim to assess the capacity of network to handle high volumes of traffic while each trial. Along with throughput other measurements made are client connection times, Station 4-Way Handshake time, DHCP times, and more. The expected behavior is for the AP to be able to handle several stations (within the limitations of the AP specs) and make sure all Clients get a fair amount of airtime both upstream and downstream. An AP that scales well will not show a significant overall throughput decrease as more Real clients are added.

py-scripts/lf_json_api.py

This script will is an example of using LANforge JSON API to use GET Requests to LANforge.

py-scripts/lf_json_util.py

This file contains a helper module standardize_json_results. standardize_json_results takes a dict of information retrieved from json_get and standardizes it to use the plural version of the data requested. The data is returned starting with the "endpoints" The script will read column data from lanforge GUI using request

py-scripts/lf_kpi_csv.py

This module is included to assist in filling out the kpi.csv correctly. The kpi.csv is used for graphing data over multiple test runs. The Unit test is used for helping to become familiar with the library

py-scripts/lf_logger_config.py

This program is a helper class for setting up python logger This helper is used by most of the LANforge python scripts.

py-scripts/lf_macvlan.py

Utility script for creating MAC vlans and setting them up and down. Can create multiple mac vlans.

py-scripts/lf_mesh_test.py

This script is used to automate running Mesh tests. You may need to view a Mesh test configured through the GUI to understand the options and how best to input data.

py-scripts/lf_mixed_traffic.py

Mixed traffic test is designed to measure the access point performance and stability by running multiple traffic on both virtual & real clients like Android, Linux, Windows, and IOS connected to the access point. This test allows the user to choose multiple types of traffic like client PING test, QOS test, FTP test, HTTP test, and multicast test. The test will create virtual stations, create CX based on the selected test, run traffic and generate a report.

py-scripts/lf_modify_radio.py

This script is designed to modify/adjust the settings of the radio's standard configuration. It allows you to easily modify basic settings like the Country code, Channel/Frequency, Antenna, TX Power, and more. Additionally, the script can activate or deactivate various features such as Extra TxStatus and Extra RxStatus for the specified radio.

py-scripts/lf_multipsk.py

This script is to test the multipsk feature in access point. Multipsk feature states connecting clients using same ssid but different passwords , here we will create two or 3 passwords with different vlan id on single ssid and try to connect client with different passwords.

py-scripts/lf_pcap.py

Common Library for reading pcap files and check packet information for specific filters.

SETUP: This script requires pyshark and tshark to be installed before

py-scripts/lf_ping_sweep.py

This Ping Sweep Test script is designed to discover devices within the network connectivity by measuring latency. It also detects issues like client unavailability time, and average latency, ensuring effective device communication and identifying connectivity problems

py-scripts/lf_report.py

This script is designed to generate reports in file formats such as PDF and HTML, accommodating various

user preferences. The reports can encompass a range of elements, including graphs, tables, and customizable objectives, tailored to meet specific user requirements

py-scripts/lf_report_test.py

This script is useful to test the functionality of the lf_graph and lf_report modules and generates pdf and html reports

py-scripts/lf_rf_char.py

The purpose of this script is to do RF Characteristics Test. This script is a work in progress put on hold.

py-scripts/lf_rfgen_info.py

This script will read the configuration settings of the RF-Generators connected to a LANforge.

py-scripts/lf_roam_test.py

The script is designed to support both hard and soft roaming, ensuring a smooth transition for devices between access points (APs). Additionally, the script captures packets in two scenarios: when a device is connected to an AP and when it roams from one AP to another. These captured packets help analyze the performance and stability of the roaming process. In essence, the script serves as a thorough test for assessing how well APs handle roaming and the overall network stability when clients move between different access points. The roaming test will create stations with advanced/802.1x and 11r key management, create CX traffic between upstream port and stations, run traffic and generate a report.

py-scripts/lf_rssi_process.py

Module to Process the data that was measured during lf_rssi_check.py , the process will take in a list of csv files extract the data and graph

py-scripts/lf_rvr_test.py

Example report: rate_vs_range.pdf

The Candela Rate vs Range Test measures the performance over distance of the Device Under Test. Distance is emulated using programmable attenuation and a throughput test is run at each distance/RSSI step and plotted on a chart. The test allows the user to plot RSSI curves both upstream and downstream for different types of traffic and different station types.

py-scripts/lf_rx_sensitivity_test.py

This script is used to automate running RX Sensitivity tests. You may need to view a RX Sensitivity test configured through the GUI to understand the options and how best to input data.

py-scripts/lf_setup_radius_server.py

lf_setup_radius_server.py will add user to users in Radius server for TTLS.

py-scripts/lf_sniff_radio.py

This script is intended to sniff the radio specified by the user on a particular channel for a specified duration.

py-scripts/lf_snp_test.py

This script is functional and is an example. This program is to test an AP connected to a controller. The AP name is configurable. The controler with with a specific ap mode, wifi mode (2.4 Ghz or 5 Ghz), Bandwidth (20,40,80,160) and TX power. This currently only works with certain models of Cisco controllers.

py-scripts/lf_test_generic.py

lf_test_generic.py will create stations and endpoints to generate traffic based on a command-line specified command type.

This script will create a variable number of stations to test generic endpoints. Multiple command types can be tested including ping, speedtest, lfcurl, iperf, generic types. The test will check the last-result attribute for different things depending on what test is being run. Ping will test for successful pings, speedtest will test for download speed, upload speed, and ping time, generic will test for successful generic commands.

This script also *does not* use any other file except lanforge_api.py.

py-scripts/lf_test_max_association.py

The Maximum Client Association Test is designed to test the capability of a newly built LANforge system. The objective of the test is to create the maximum number of virtual station interfaces on the system installed WIFI radio, associate the stations to the specified AP and run a long duration layer-3 UDP bidirectional traffic test. The test will create stations, create CX traffic & run traffic specified time period and generate a report.

py-scripts/lf_tr398_test.py

This script is used to automate running TR398 tests. You may need to view a TR398 test configured through the GUI to understand the options and how best to input data. (Issue 1)

py-scripts/lf_tr398v2_test.py

This script is used to automate running TR398v2 tests. You may need to view a TR398v2 test configured through the GUI to understand the options and how best to input data.

py-scripts/lf_tr398v4_test.py

Automate running TR398 issue 4 tests. See cv_examples/run_tr398_71.bash for example of how to use this in a larger context.

py-scripts/lf_webpage.py

lf_webpage.py will verify that N clients are connected on a specified band and can download some amount of file data from the HTTP server while measuring the time taken by clients to download the file and number of times the file is downloaded.

py-scripts/lf_we_can_wifi_capacity.py

./lf_we_can_wifi_capacity.py is used for running Wi-Fi capacity test on real clients (Phones).

py-scripts/lf_wifi_capacity_test.py

Example report: wifi_capacity.pdf

The Candela WiFi Capacity test is designed to measure performance of an Access Point when handling different amounts of WiFi Stations. The test allows the user to increase the number of stations in user defined steps for each test iteration and measure the per station and the overall throughput for each trial. Along with throughput other measurements made are client connection times, Fairness, % packet loss, DHCP times and more. The expected behavior is for the AP to be able to handle several stations (within the limitations of the AP specs) and make sure all stations get a fair amount of airtime both in the upstream and downstream. An AP that scales well will not show a significant over-all throughput decrease as more stations are added.

py-scripts/measure_station_time_up.py

./measure_station_time_up.py is for creating a variable number of stations and measures the time it took to admin them up and get IP addresses.

py-scripts/modify_station.py

Modify stations on a system. Use the enable_flag to create a flag on a station. Turn off a flag with the disable_flag option. A list of available flags are available in the add_station.py file in py-json/LANforge.

py-scripts/modify_vap.py

./modify_vap.py will modify VAPs on a system. Use the enable_flag to create a flag on a VAP. Turn off a flag with the disable_flag option. A list of available flags are available in the add_vap.py file in py-json/LANforge.

py-scripts/monitor_cx.py

The Monitor CX script is for collecting CSV data from running Layer 3 connections. It reports data for each endpoint. Start this script after beginning traffic in the GUI or other means. When all connections have stopped, this script will exit. If this script is run before connections are started, it will immediately exit.

py-scripts/raw_cli.py

Utility script intended to be used from shell scripts in order to send commands to a LANforge system through the REST API. This script can send a one-line preformatted command like the kind found in a /home/lanforge/DB directory, or can assemble a command using arguments.

py-scripts/run_cv_scenario.py

Chamber View testing script: Load a Chamber View (CV) scenario, build it, and run a test.

py-scripts/run_voip_cx.py

This script will start a named set of voip connections and report their data to a csv file

py-scripts/rvr_scenario.py

LANforge Reporting Script: Load a scenario and run a RvR report.

py-scripts/scenario.py

The script, scenario.py, will load a database file and control test groups.

py-scripts/ssh_remote.py

This script is an example do_ap script on remote system.

py-scripts/sta_connect2.py

Test will create a station, create TCP and UDP traffic, run it a short amount of time, and verify whether traffic was sent and received. It also verifies the station connected to the requested BSSID if bssid is specified as an argument. The script will clean up the station and connections at the end of the test.

py-scripts/sta_connect_bssid_mac.py

Work in progress. The script can create stations and can be used to set multiple BSSID, MAC to each individual station.

py-scripts/sta_connect_example.py

This script is no longer supported: Example of how to instantiate StaConnect and run the test.

py-scripts/sta_connect_multi_example.py

This script is an example of how to instantiate StaConnect and run the test.

py-scripts/sta_scan_test.py

This scripts optionally creates a station with specified ssid info (can be real or fake ssid, if fake use open for security). If not creating a station, it can use existing station. Then starts a scan and waits 15 seconds, finally scan results are printed to console.

py-scripts/stations_connected.py

This script will return the count of the number of stations connected to a specific bssid

py-scripts/test_client_admission.py

Script not functional or no longer supported: This script will create one station at a time and generate downstream traffic at 5Mbps then again create next station create layer3 and will continue doing same until Ap stops admiting client. This script can be used for for client admission test for particular AP

py-scripts/test_fileio.py

test_fileio.py will create stations or macvlans with matching fileio endpoints to generate and verify fileio related traffic.

This script will create a variable number of stations or macvlans to test fileio traffic. Pre-existing stations and macvlans can be used as well. Command line options are available to update cross-connects as well as using a list of existing cross-connects if desired. if none are given, cross-connects and endpoints will be created by the script. Modes such as read-only, write-only, or both can be specified along with ip addresses and starting numbers for sequential stations or macvlans that are created in case of limited or pre-existing configurations. The test that is run during this script will depend on the mode used, a read-only test will check the read-bps attribute, write-only will check write-bps and both will check both attributes. If the relevant attributes increase over the duration of the test it will pass, otherwise it will fail.

py-scripts/test_generic.py

test_generic.py will create stations and endpoints to generate traffic based on a command-line specified command type.

This script will create a variable number of stations to test generic endpoints. Multiple command types can be tested including ping, speedtest, generic types. The test will check the last-result attribute for different things depending on what test is being run. Ping will test for successful pings, speedtest will test for download speed, upload speed, and ping time, generic will test for successful generic commands

py-scripts/testgroup.py

This script is designed to create a test connection group in the LANforge GUI (Connection Group GUI tab). Test Groups, also known as Connection Groups, can be created, modified, and managed using this script. You can use it to create a new test group, add or remove layer-3 connections from the group, and start or stop the entire test connection group as needed. It simplifies the process of handling these tasks within the LANforge environment.

py-scripts/test_ip_connection.py

./test_ip_connection.py may not be functional , added this summary to allow the help_summary check to pass The script si create station then attemp to authenticate, associate, and receive IP address on the chosen SSID

py-scripts/test_ipv4_ps.py

This script test ipv4 ps , this script functionality has not been validated recently as of 12/28/2024 Help summary is put in to allow help_summary test to pass

py-scripts/test_ipv4_ttls.py

This script test ipv4 ttls , this script functionality has not been validated recently as of 12/28/2024 Help summary is put in to allow help_summary test to pass

py-scripts/test_ip_variable_time.py

This script is designed to generate a variable number (N) of stations and establish cross connections while facilitating the execution of UDP/TCP layer 3 traffic for a duration specified by the user. Additionally, it supports the utilization of existing stations and the creation of stations across multiple radios. Furthermore, stations can connect to multiple SSIDs as specified by the user.

py-scripts/test_l3_longevity.py

The Layer 3 Traffic Generation Test is designed to test the performance of the Access Point by running layer 3 TCP and/or UDP Traffic. Layer-3 Cross-Connects represent a stream of data flowing through the system under test. A Cross-Connect (CX) is composed of two Endpoints, each of which is associated with a particular Port (physical or virtual interface).

The test will create stations, create CX traffic between upstream port and stations, run traffic and generate a report.

py-scripts/test_l3_powersave_traffic.py

Script not functional or no longer supported: see script py-scripts/test_l3.py as replacement

py-scripts/test_l3.py

Example report: test_l3.pdf

The Layer 3 Traffic Generation Test is designed to test the performance of the Access Point by running layer 3 TCP and/or UDP Traffic. Layer-3 Cross-Connects represent a stream of data flowing through the system under test. A Cross-Connect (CX) is composed of two Endpoints, each of which is associated with a particular Port (physical or virtual interface).

The test will create stations, create CX traffic between upstream port and stations, run traffic and generate a report.

py-scripts/test_l3_scenario_throughput.py

Script not functional or no longer supported: Test Scenario of DUT Temperature measurement along with simultaneous throughput on VAP as well as stations

py-scripts/test_l3_unicast_traffic_gen.py

Script Deprecated: This script is replaced by py-scripts/test_l3.py.

py-scripts/test_l3_WAN_LAN.py

Work in progress. This script is for testing WAN to LAN traffic.

py-scripts/test_l4.py

Work in Progress: This script will create stations and endpoints to generate and verify layer-4 traffic by monitoring the urls/s, bytes-rd, or bytes-wr attribute of the endpoints.

py-scripts/test_status_msg.py

This scripts will test the status message passing functions of /status-msg: - create a session: PUT /status-msg/ - post message: POST /status-msg/ - list sessions: GET /status-msg/ - list messages for session: GET /status-msg/ - delete message: DELETE /status-msg//message-id - delete session: DELETE /status-msg//this - delete all messages in session: DELETE /status-msg//all

py-scripts/test_wanlink.py

This script will create and modify WAN Links from the command line.

py-scripts/throughput_qos.py

The Throughput QoS test is designed to measure performance of an Access Point while running traffic with different types of services like voice, video, best effort, background. The test allows the user to run layer3 traffic for different ToS in upload, download and bi-direction scenarios between AP and virtual devices. Throughputs for all the ToS are reported for individual clients along with the overall throughput for each ToS. The expected behavior is for the AP to be able to prioritize the ToS in an order of voice,video,best effort and background. The test will create stations, create CX traffic between upstream port and stations, run traffic and generate a report.

py-scripts/tip_station_powersave.py

This script uses filters from realm's PacketFilter class to filter pcap output for specific packets. Currently it uses a filter for association packets using wlan.fc.type_subtype<=3. It is also using a filter for QOS Null packets using wlan.fc.type_subtype==44. Both filters are also looking for the existence of either the station MAC or the AP MAC in wlan.addr These are returned as an array of lines from the output in the format $subtype $mac_addresses $wlan.fc.pwrmgt

py-scripts/update_dependencies.py

The lanforge-scripts/py-scripts and lanforge-scripts/py-json collection require a number of Pypi libraries. This script installs those libraries or creates a virtual environment for those libraries. This script has been updated to detect PEP 668 externally-managed libraries; in the presence of the externally-managed condition, a virtual environment will be created in $home/scripts/venv-$3.8. This script will update a symlink $home/lanforge/venv to default virtual environment as necessary.

py-scripts/webGUI_update_dependencies.py

This script installs python3 webGUI package dependencies.

py-scripts/wlan_capacity_calculator.py

This script calculates the theoretical value of three different stations(11abg/11n/11ac)

Script Information

py-scripts/attenuator_serial.py

```
usage: attenuator_serial.py [-h] [--help_summary]

attenuator_serial.py: this file is used in tip for getting serial number of attenuators,


optional arguments:
```

```
  -h, --help      show this help message and exit
  --help_summary  Show summary of what this script does

 Useful Information:
 this file is used in tip for getting serial number of attenuators,
```

py-scripts/bandsteering.py

```
usage: bs_obj.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [-u UPSTREAM_PORT]
                 [--num_stations NUM_STATIONS] [--test_id TEST_ID] [-d]
                 [--log_level LOG_LEVEL]
                 [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                 [--proxy [PROXY]] [--debugging DEBUGGING [DEBUGGING ...]]
                 [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
                 [--radio RADIO] [--security SECURITY] [--ssid SSID]
                 [--passwd PASSWD] [--radio_2g RADIO_2G] [--radio_5g RADIO_5G]
                 [--a_min A_MIN] [--b_min B_MIN] [--iter ITER]
                 [--traffic_type TRAFFIC_TYPE]

       bs_obj.py
--------------------
Command example:
./bs_obj.py
    --radio wiphy0
    --radio_2g wiphy1
    --radio_5g wiphy2
    --num_stations 3
    --security open
    --ssid netgear
    --passwd BLANK
    --debug


optional arguments:
  -h, --help            show this help message and exit
  --radio_2g RADIO_2G   radio for 2.4 Ghz
  --radio_5g RADIO_5G   radio for 5 Ghz
  --a_min A_MIN         --a_min bps rate minimum for side_a
  --b_min B_MIN         --b_min bps rate minimum for side_b
  --iter ITER           --iter is used to provide number of iterations
  --traffic_type TRAFFIC_TYPE
                        traffic type tcp or udp,for tcp lf_tcp and udp lf_udp

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

        Create stations
```

py-scripts/bssid_to_dut.py

```
usage: bssid_script.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                       [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                       [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                       [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                       [--proxy [PROXY]]
                       [--debugging DEBUGGING [DEBUGGING ...]]
                       [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
                       [--radio RADIO] [--security SECURITY] [--ssid SSID]
                       [--passwd PASSWD] [--use_existing_station]
                       [--mode MODE] [--sta_name STA_NAME [STA_NAME ...]]
                       [--csv_output CSV_OUTPUT] [--scan_time SCAN_TIME]
                       [--dut_name DUT_NAME]

       Creates a temporary station with specified ssid info (ssid, security, password)
       Then starts a scan, waits 15 seconds, and prints scan results to console.
       Takes 2 BSSIDs of ssids (given to temp station) and creates DUT (named what is given in dut_name argument).

       TODO: take scan results and calculate how many unique BSSIDs are there from the scan results, and add that many ssids to t
       TODO: parse scan results for more than 1 SSID name (ssid-2G, ssid-5G). Add both ssids to the dut_name.
```

```
       Example:
       ./bssid_to_dut.py --ssid test_name --security open --radio wiphy0 --dut_name lanforge-AP


optional arguments:
  -h, --help            show this help message and exit
  --use_existing_station
                        Use existing station instead of trying to create stations.
  --mode MODE           Used to force mode of stations
  --sta_name STA_NAME [STA_NAME ...]
                        Optional: User defined station names: 1.2.wlan0 1.3.wlan0
  --csv_output CSV_OUTPUT
                        Specify file to which csv output will be saved, otherwise print it in the terminal
  --scan_time SCAN_TIME
                        Specify time in seconds to wait for scan to complete.  Default is 15
  --dut_name DUT_NAME   Specify the name the new DUT should be called. Default name is bssid_dut.

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

       Used to scan for ssids after creating a station
```

py-scripts/chamber_ctl.py

```
usage: ./chamber_ctl.py [-h] [--host HOST] [--debug] [--chamber CHAMBER]
                        [--info] [--position POSITION]
                        [--adjust_position ADJUST_POSITION] [--tilt TILT]
                        [--adjust_tilt ADJUST_TILT] [--speed SPEED]
                        [--log_level LOG_LEVEL] [--no_settle] [--help_summary]

Operate a LANForge-connected turntable. NOTE: "--position" and "--tilt" will override their adjust argument counterparts

optional arguments:
  -h, --help            show this help message and exit
  --host HOST, --mgr HOST
                        specify the GUI to connect to, assumes port 8080
  --debug               turn on debugging
  --chamber CHAMBER     Chamber name (e.g. Chamber-0). Turntables are features of ChamberView chambers. There can be multiple reso
  --info                Display chamber turntable information
  --position POSITION   Set the turn table position to an absolute position (between 0.0 and 359.9). Position resolution is 1/10th
  --adjust_position ADJUST_POSITION
                        Adjust the turn table position a few degrees relative to current position. Negative degrees turn the table
  --tilt TILT           Set the turn table tilt to an absolute position (between 0.0 and 359.9). Position resolution is 1/10th of
  --adjust_tilt ADJUST_TILT
                        Adjust the turn table tilt a few degrees relative to current position. Requires 845D turntable. See "--adj
  --speed SPEED, --rpm SPEED
                        Turn table rotation speed, in RPM. Minimum is 0.1 RPM, maximum is 7 RPM
  --log_level LOG_LEVEL
                        Set log level
  --no_settle           Exit script before turntable reports no movement for more than two pollings.
  --help_summary        Show summary of what this script does
```

py-scripts/create_bond.py

```
usage: create_bond.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                      [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                      [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                      [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                      [--proxy [PROXY]]
                      [--debugging DEBUGGING [DEBUGGING ...]]
                      [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
                      [--radio RADIO] [--security SECURITY] [--ssid SSID]
                      [--passwd PASSWD] [--bond_name BOND_NAME]
                      [--network_dev_list NETWORK_DEV_LIST]

NAME:       create_bond.py

PURPOSE:    Create and configure a single bond port using the specified child ports.

EXAMPLE:    # Bond with two child bridge ports. No IPv4 configuration specified.
               ./create_bond.py                    --bond_name         bond0                    --network_dev_list  eth1,eth2
```

```
VERIFIED_ON:
           Tested on 03/22/2023:
           Kernel version: 5.19.17+
           GUI version: 5.4.6

LICENSE:    Free to distribute and modify. LANforge systems must be licensed.
           Copyright 2024 Candela Technologies Inc


optional arguments:
  -h, --help            show this help message and exit

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

required arguments:
  --bond_name BOND_NAME
                        Name of the bridge to create
  --network_dev_list NETWORK_DEV_LIST
                        List of network devices in the bond, must be comma separated with no spaces

        Create bonds
```

py-scripts/create_bridge.py

```
usage: create_bridge.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                        [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                        [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                        [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                        [--proxy [PROXY]]
                        [--debugging DEBUGGING [DEBUGGING ...]]
                        [--debug_log DEBUG_LOG] [--no_cleanup]
                        [--help_summary] [--radio RADIO] [--security SECURITY]
                        [--ssid SSID] [--passwd PASSWD]
                        [--bridge_name BRIDGE_EID]
                        [--bridge_ports BRIDGE_PORTS [BRIDGE_PORTS ...]]
                        [--cleanup] [--create_admin_down]
                        [--dhcp | --ip IPV4_ADDRESS] [--netmask IPV4_NETMASK]
                        [--gateway IPV4_GATEWAY]

NAME:       create_bridge.py

PURPOSE:    Create and configure a single bridge port using the specified child ports.

NOTES:      WiFi station interfaces are not supported as a child port.

            This script will optionally set IPv4 configuration (static or dynamic), if specified.
            The specified IPv4 configuration will be set for the bridge port when specified.
            Child bridge port IPv4 configuration will always be cleared (TODO).

EXAMPLE:    # Bridge with two child bridge ports. No IPv4 configuration specified.
                ./create_bridge.py                    --bridge_name   1.1.br0                   --bridge_ports  1.1.eth2,1.1.et

            # Bridge with two child bridge ports. No IPv4 configuration specified.
            # Assumes bridge created on resource 1 (assumed from bridge child ports).
                ./create_bridge.py                    --bridge_name   br0                    --bridge_ports  1.1.eth2,1.1.eth3

            # Bridge with two child bridge ports created in down state. No IPv4 configuration specified.
                ./create_bridge.py                    --bridge_name   1.1.br0                   --bridge_ports  1.1.eth2,1.1.et
                    --create_admin_down

            # Bridge with two child bridge ports created. DHCPv4 enabled.
                ./create_bridge.py                    --bridge_name   1.1.br0                   --bridge_ports  1.1.eth2,1.1.et
                    --dhcpv4

            # Bridge with two child bridge ports created. Static IPv4 configuration.
                ./create_bridge.py                    --bridge_name   1.1.br0                   --bridge_ports  1.1.eth2,1.1.et
                    --ipv4_address  172.16.0.10                   --ipv4_netmask  255.255.255.0                    --ipv4_gatew

LICENSE:    Free to distribute and modify. LANforge systems must be licensed.
            Copyright 2024 Candela Technologies Inc.
```

```
optional arguments:
  -h, --help            show this help message and exit
  --bridge_name BRIDGE_EID
                        Name of the bridge port to create. This can be either the name only or the full EID. If not the full EID,
  --bridge_ports BRIDGE_PORTS [BRIDGE_PORTS ...], --target_device BRIDGE_PORTS [BRIDGE_PORTS ...]
                        Ports to bridge together. If not specified, in the '--bridge_name' argument, the resource ID will be infer
  --cleanup             Clean up any created ports before exiting.
  --create_admin_down   Create bridge port in admin down state. Will not attempt to admin up bridged ports.
  --dhcp, --dhcpv4, --use_dhcp
                        Enable DHCPv4 on created bridge port
  --ip IPV4_ADDRESS, --ipv4_address IPV4_ADDRESS
                        Set static IPv4 address for the created bridge port
  --netmask IPV4_NETMASK, --ipv4_netmask IPV4_NETMASK
                        IPv4 subnet mask to apply to created bridge port when static IPv4 configuration requested
  --gateway IPV4_GATEWAY, --ipv4_gateway IPV4_GATEWAY
                        IPv4 gateway to apply to created bridge port when static IPv4 configuration requested

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

        Create bridges
```

py-scripts/create_chamberview_dut.py

```
usage: create_chamberview_dut.py [-h] [-m LFMGR] [-o PORT] [-d DUT_NAME]
                                 [-s SSID] [--sw_version SW_VERSION]
                                 [--hw_version HW_VERSION]
                                 [--serial_num SERIAL_NUM]
                                 [--model_num MODEL_NUM] [--dut_flag DUT_FLAG]
                                 [--dut_notes DUT_NOTES] [--dut_notes_clear]
                                 [--debug] [--log_level LOG_LEVEL]
                                 [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                 [--help_summary]

NAME: create_chamberview_dut.py

PURPOSE: This script creates a dut with given parameters like dut name,ssid,security,bssid in chamber view.

EXAMPLE:
        ./create_chamberview_dut --lfmgr localhost --mgr_port 8080 --dut_name dut_name
        --ssid "ssid_idx=0 ssid=NET1 security=WPA|WEP|11r|EAP-PEAP|open bssid=78:d2:94:bf:16:41"
        --ssid "ssid_idx=1 ssid=NET1 security=WPA password=test bssid=78:d2:94:bf:16:40"

SCRIPT_CLASSIFICATION :  Creation

SCRIPT_CATEGORIES:   Functional

NOTES:
        To Run this script gui should be opened with

            path: cd LANforgeGUI_5.4.3 (5.4.3 can be changed with GUI version)
                  pwd (Output : /home/lanforge/LANforgeGUI_5.4.3)
                  ./lfclient.bash -cli-socket 3990

        This script is used to create a DUT in chamber view.
                Manual steps:
                   1. open GUI
                   2. click Chamber View
                   3. right click on empty space in Scenario configuration  select "New DUT"
                   4. Enter Name (DUT Name), SSID , Security type, BSsid (if available)
                   5. click on apply and OK
                   6. you will see a DUT created in chamber view under scenario configuration

        If entered DUT name is already created in lanforge,
        it will overwrite on to that DUT ( All information will be overwritten )
        Which means it will "Update the DUT".

        If entered DUT name is not already in lanforge,
        then new DUT will be created will all the provided information

        The contents of '--ssid' argument are split with shlex, so you can do commands like this as well:
```

```
        ./create_chamberview_dut.py --lfmgr localhost --dut_name regression_dut
            --ssid "ssid_idx=0 ssid='j-wpa2-153 space' security='wpa2' password=j-wpa2-153 bssid=04:f0:21:cb:01:8b"

        --lfmgr = IP of lanforge
        --mgr_port = Default 8080
        --dut_name = Enter name of DUT ( to update DUT enter same DUT name )
                                    ( enter new DUT name to create a new DUT)
        --ssid = "ssid_idx=0 ssid=NET1 security=WPA|WEP|11r|EAP-PEAP bssid=78:d2:94:bf:16:41"

                --ssid will take = ssid_idx (from 0 to 7) : we can add upto 7 ssids to a DUT
                                    = ssid : Name of SSID
                                    = security : Security type WPA|WEP|11r|EAP-PEAP ( in case of multiple security add "|"
                                            after each type ex. WPA|WEP (this will select WPA and WEP both)
                                    = bssid : Enter BSSID
                                    (if you dont want to give bssid
                                    --ssid "ssid_idx=0 ssid=NET1 security=WPA|WEP|11r|EAP-PEAP"
                                    )

    STATUS: Functional

    VERIFIED_ON: 31-JULY-2023
                Build version - 5.4.6
                kernel version - 6.2.16+

    LICENSE:
            Free to distribute and modify. LANforge systems must be licensed.
            Copyright 2023 Candela Technologies Inc

    INCLUDE_IN_README: False

    optional arguments:
      -h, --help            show this help message and exit
      -m LFMGR, --lfmgr LFMGR, --mgr LFMGR
                            address of the LANforge GUI machine (localhost is default)
      -o PORT, --port PORT, --mgr_port PORT
                            IP Port the LANforge GUI is listening on (8080 is default)
      -d DUT_NAME, --dut_name DUT_NAME
                            set dut name
      -s SSID, --ssid SSID  SSID
      --sw_version SW_VERSION
                            DUT Software version.
      --hw_version HW_VERSION
                            DUT Hardware version.
      --serial_num SERIAL_NUM
                            DUT Serial number.
      --model_num MODEL_NUM
                            DUT Model Number.
      --dut_flag DUT_FLAG   DUT flags to add
      --dut_notes DUT_NOTES
                            Add Notes to Chamberview Test, may want to use --dut_notes_clear prior
      --dut_notes_clear     Clear out older notes, used prior to adding new notes will set '[BLANK]'
      --debug               Enable debugging
      --log_level LOG_LEVEL
                            Set logging level: debug | info | warning | error | critical
      --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                            --lf_logger_config_json <json file> , json configuration of logger
      --help_summary        Show summary of what this script does
```

py-scripts/create_chamberview.py

```
    usage: create_chamberview.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                                 [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                                 [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                                 [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                 [--proxy [PROXY]]
                                 [--debugging DEBUGGING [DEBUGGING ...]]
                                 [--debug_log DEBUG_LOG] [--no_cleanup]
                                 [--help_summary] [--radio RADIO]
                                 [--security SECURITY] [--ssid SSID]
                                 [--passwd PASSWD] [-cs CREATE_SCENARIO]
                                 [-l LINE [LINE ...]] [-rl RAW_LINE] [-ds]

    NAME: create_chamberview.py

    PURPOSE: This script creates a scenario in which stations,bridged-AP,vap,etc can be created and upstream, upstream-dhcp,
             uplink-nat can be configured in chamber view.

    EXAMPLE:
            EXAMPLE-1:
            ./create_chamberview.py -m "localhost" -o "8080" -cs "scenario_name"
                --line "Resource=1.1 Profile=STA-AC Amount=1 Uses-1=wiphy0 Uses-2=AUTO Freq=-1
                    DUT=Test DUT_Radio=Radio-1 Traffic=http VLAN=NA"
                --line "Resource=1.1 Profile=upstream Amount=1 Uses-1=eth1 Uses-2=AUTO Freq=-1
                    DUT=Test DUT_Radio=Radio-1 Traffic=http VLAN=NA"

            EXAMPLE-2:
            ./create_chamberview.py -m "localhost" -o "8080" -cs "scenario_name"
                --raw_line "profile_link 1.1 STA-AC 10 'DUT: temp Radio-1' tcp-dl-6m-vi wiphy0,AUTO -1"
                --raw_line "profile_link 1.1 upstream 1 'DUT: temp Radio-1' tcp-dl-6m-vi eth1,AUTO -1"

    SCRIPT_CLASSIFICATION :  Creation

    SCRIPT_CATEGORIES:   Functional

    NOTES:
            To Run this script gui should be opened with

                path: cd LANforgeGUI_5.4.3 (5.4.3 can be changed with GUI version)
                      pwd (Output : /home/lanforge/LANforgeGUI_5.4.3)
                      ./lfclient.bash -cli-socket 3990

            Scenario names should be different, for each run of this script.
                in case of same scenario name scenario will be appended to the same name.

            You should see build scenario with the given arguments at the end of this script.
                To verify this:
                    open Chamber View -> Manage scenario
```

```
STATUS: Functional

VERIFIED_ON: 31-JULy-2023
             Build version - 5.4.6
             kernel version - 6.2.16+

LICENSE:
        Free to distribute and modify. LANforge systems must be licensed.
        Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False

optional arguments:
  -h, --help            show this help message and exit
  -cs CREATE_SCENARIO, --create_scenario CREATE_SCENARIO, --create_lf_scenario CREATE_SCENARIO
                        name of scenario to be created and saved in lanforge database. See further notes section for more details.
  -l LINE [LINE ...], --line LINE [LINE ...]
                        line number
  -rl RAW_LINE, --raw_line RAW_LINE
                        raw lines
  -ds, --delete_scenario
                        delete existing scenario with same name as given in 'create-scenario' argument. See 'further notes' sectio

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key
```

py-scripts/create_l3.py

```
usage: create_l3.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [--debug]
                    [--log_level LOG_LEVEL]
                    [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                    [--proxy [PROXY]] [--help_summary] [--cleanup]
                    [--no_cleanup] [--no_pre_cleanup] [--cx_prefix CX_PREFIX]
                    [--endp_a ENDP_A] [--endp_b ENDP_B]
                    [--ep_pairs [EP_PAIRS [EP_PAIRS ...]]]
                    [--min_ip_port_a MIN_IP_PORT_A]
                    [--min_ip_port_b MIN_IP_PORT_B] [--cx_type CX_TYPE]
                    [--min_rate_a MIN_RATE_A] [--min_rate_b MIN_RATE_B]
                    [--tos TOS] [--pkts_to_send PKTS_TO_SEND]
                    [--multi_conn_a MULTI_CONN_A]
                    [--multi_conn_b MULTI_CONN_B]
                    [--batch_quantity BATCH_QUANTITY]
                    [--endp_a_increment ENDP_A_INCREMENT]
                    [--endp_b_increment ENDP_B_INCREMENT]
                    [--ip_port_increment_a IP_PORT_INCREMENT_A]
                    [--ip_port_increment_b IP_PORT_INCREMENT_B]

NAME: create_l3.py

PURPOSE: This script is used to create the user-specified Layer-3 cross-connection.

EXAMPLE:
        (If the specified endpoints are not present in the port manager,the cross-connects will be in a PHANTOM state.)

        # For layer-3 cx creation on LANforge:

            ./create_l3.py --mgr localhost --endp_a eth1 --endp_b eth2 --min_rate_a 56000 --min_rate_b 40000 --no_cleanup

        # For regression (script will create the layer-3 cx, check if it was successful, and then remove the layer-3 cx):

            ./create_l3.py --mgr localhost --endp_a 1.1.sta0000 --endp_b 1.2.sta0000 --min_rate_a 56000 --min_rate_b 40000 --no_cl

        # For remote layer-3 cx creation:

            ./create_l3.py --mgr localhost --endp_a sta0000,sta0001 --endp_b eth2 --min_rate_a 56000 --min_rate_b 40000 --cx_type
             --multi_conn_a 1 --multi_conn_b 1 --no_cleanup

        # With tos & pkts_to_send cross-connections

            ./create_l3.py --mgr localhost --endp_a sta00 --endp_b eth2 --min_ip_port_a 1000 --tos VI --pkts_to_send 10 --no_clean

        # For batch creation functionality:
```

```
           ./create_l3.py --mgr 192.168.200.93 --endp_a 1.1.eth1 --endp_b 1.1.wlan2 --min_rate_a 6200000 --min_rate_b 6200000
            --batch_quantity 10 --endp_a_increment 0 --endp_b_increment 1 --min_ip_port_a 1000 --min_ip_port_b 2000
            --ip_port_increment_a 1 --ip_port_increment_b 1 --multi_conn_a 1 --multi_conn_b 1 --no_cleanup

SCRIPT_CLASSIFICATION:  Creation

SCRIPT_CATEGORIES:   Functional

NOTES:
        Create Layer-3 Cross Connection Using LANforge JSON API : https://www.candelatech.com/cookbook.php?vol=fire&book=scripted+
        Written by Candela Technologies Inc.

        * Supports only creating user-specified Layer-3 cross-connection.
        * Supports regression testing for QA

        * If the specified ports used for creating endpoints are not present in the port manager, the cross-connects will be in a

STATUS: Functional

VERIFIED_ON:   20-JUN-2023,
            Build Version:  5.4.6
            Kernel Version: 6.2.14+

LICENSE:
         Free to distribute and modify. LANforge systems must be licensed.
         Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False

optional arguments:
  -h, --help            show this help message and exit
  --cleanup             Cleanup before exit
  --no_cleanup          Deprecated option. This script does not cleanup by default.
  --no_pre_cleanup      do not remove connections at start
  --cx_prefix CX_PREFIX
                        phrase to begin CX names with
  --endp_a ENDP_A       --endp_a station list
  --endp_b ENDP_B       --upstream port
  --ep_pairs [EP_PAIRS [EP_PAIRS ...]]
                        --ep_pairs is a list of ports in a,b format, like eth1#0,eth2#0 eth1#1,eth2#1
  --min_ip_port_a MIN_IP_PORT_A
                        min ip port range for endp-a
  --min_ip_port_b MIN_IP_PORT_B
                        min ip port range for endp-b
  --cx_type CX_TYPE     specify the traffic type for cx eg : lf_udp | lf_tcp
  --min_rate_a MIN_RATE_A
                        --min_rate_a bps rate minimum for side_a
  --min_rate_b MIN_RATE_B
                        --min_rate_b bps rate minimum for side_b
  --tos TOS             specify tos for endpoints eg : BK | BE | VI | VO | Voice | Video
  --pkts_to_send PKTS_TO_SEND
                        specify the pkts to send to the endpoints eg :One - 1 | Ten - 10 | (100) - 100 | (1000) - 1000
  --multi_conn_a MULTI_CONN_A, --multi_con_a MULTI_CONN_A
                        modify multi connection endpoint-a for cx
  --multi_conn_b MULTI_CONN_B, --multi_con_b MULTI_CONN_B
                        modify multi connection endpoint-b for cx
  --batch_quantity BATCH_QUANTITY
                        No of cx endpoints to batch-create
  --endp_a_increment ENDP_A_INCREMENT
                        End point - A port increment
  --endp_b_increment ENDP_B_INCREMENT
                        End point - B port increment
  --ip_port_increment_a IP_PORT_INCREMENT_A
                        ip port increment for endp-a
  --ip_port_increment_b IP_PORT_INCREMENT_B
                        ip port increment for endp-b

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_bare_argparse found in /lanforge-scripts/py-json/LANfo
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --debug, -d           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80 or https://user:pass@proxy.localnet:3128
  --help_summary        Show summary of what this script does

           Used for creating layer-3 cross connections
```

py-scripts/create_l3_stations.py

```
usage: create_l3_stations.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                             [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                             [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                             [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                             [--proxy [PROXY]]
                             [--debugging DEBUGGING [DEBUGGING ...]]
                             [--debug_log DEBUG_LOG] [--no_cleanup]
                             [--help_summary] [--radio RADIO]
                             [--security SECURITY] [--ssid SSID]
                             [--passwd PASSWD] [--min_rate_a MIN_RATE_A]
                             [--min_rate_b MIN_RATE_B] [--mode MODE] [--ap AP]
                             [--number_template NUMBER_TEMPLATE]
                             [--station_list STATION_LIST [STATION_LIST ...]]
                             [--batch_create]
                             [--batch_quantity BATCH_QUANTITY]
                             [--endp_a ENDP_A] [--endp_b ENDP_B]
                             [--multi_conn_a MULTI_CONN_A]
                             [--multi_conn_b MULTI_CONN_B]
                             [--min_ip_port_a MIN_IP_PORT_A]
                             [--min_ip_port_b MIN_IP_PORT_B]
                             [--endp_a_increment ENDP_A_INCREMENT]
                             [--endp_b_increment ENDP_B_INCREMENT]
                             [--ip_port_increment_a IP_PORT_INCREMENT_A]
```

```
                        [--ip_port_increment_b IP_PORT_INCREMENT_B]

"""
NAME: create_l3_stations.py

PURPOSE:
         ->  This script creates variable number of stations with individual cross-connects and endpoints.
             Stations are set to UP state, but cross-connections remain stopped.

         ->  This script support Batch-create Functionality.

EXAMPLE:
        Default configuration:
            Endpoint A: List of stations (default: 2 stations, unless specified with --num_stations)
            Endpoint B: eth1

        * Creating specified number of station names and Layer-3 CX :

            ./create_l3_stations.py --mgr localhost --num_stations 5 --radio wiphy0 --ssid SSID --password Password@123 --security

        * Creating stations with specified start ID (--num_template) and Layer-3 CX :

            ./create_l3_stations.py --mgr localhost --number_template 007 --radio wiphy0 --ssid SSID --password Password@123 --sec

        * Creating stations with specified names and Layer-3 CX :

            ./create_l3_stations.py --mgr localhost --station_list sta00,sta01 --radio wiphy0 --ssid SSID --password Password@123

        * For creating stations and layer-3 cx creation on particular specified AP mac & mode:

            ./create_l3_stations.py --mgr localhost --radio wiphy0 --ssid SSID --password Password@123 --security wpa2 --ap "00:0e
            --mode 13

        * For creating specified number of stations and layer-3 cx creation (Customise the traffic and upstream port):

            ./create_l3_stations.py --mgr localhost --station_list sta00  --radio wiphy0 --ssid SSID --password Password@123 --sec
             --upstream_port eth2 --min_rate_a 6200000 --min_rate_b 6200000

        * For Batch-Create :

            ./create_l3_stations.py --mgr 192.168.200.93 --endp_a 1.1.eth2 --endp_b 1.1.sta0002 --min_rate_a 6200000 --min_rate_b
            --batch_create --batch_quantity 8 --endp_a_increment 0 --endp_b_increment 0 --min_ip_port_a 1000 --min_ip_port_b 2000
            --ip_port_increment_a 1 --ip_port_increment_b 1 --multi_conn_a 1 --multi_conn_b 1

      Generic command layout:

        python3 ./create_l3_stations.py
            --upstream_port eth1
            --radio wiphy0
            --num_stations 32
            --security {open|wep|wpa|wpa2|wpa3}
            --ssid netgear
            --password admin123
            --min_rate_a 1000
            --min_rate_b 1000
            --ap "00:0e:8e:78:e1:76"
            --number_template 0000
            --mode   1
                {"auto"   : "0",
                "a"      : "1",
                "b"      : "2",
                "g"      : "3",
                "abg"    : "4",
                "abgn"   : "5",
                "bgn"    : "6",
                "bg"     : "7",
                "abgnAC" : "8",
                "anAC"   : "9",
                "an"     : "10",
                "bgnAC"  : "11",
                "abgnAX" : "12",
                "bgnAX"  : "13",
            --debug

SCRIPT_CLASSIFICATION:  Creation

SCRIPT_CATEGORIES:   Functional

NOTES:
        Create Layer-3 Cross Connection Using LANforge JSON API : https://www.candelatech.com/cookbook.php?vol=fire&book=scripted+
        Written by Candela Technologies Inc.

        * Supports creating of stations and creates Layer-3 cross-connection with the endpoint_A as stations created and endpoint_
        * Supports regression testing for QA

STATUS: Functional

VERIFIED_ON:   27-JUN-2023,
             Build Version:  5.4.6
             Kernel Version: 6.2.14+

LICENSE:
          Free to distribute and modify. LANforge systems must be licensed.
          Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False

optional arguments:
  -h, --help            show this help message and exit
  --min_rate_a MIN_RATE_A
                        --min_rate_a bps rate minimum for side_a
  --min_rate_b MIN_RATE_B
                        --min_rate_b bps rate minimum for side_b
  --mode MODE           Used to force mode of stations
  --ap AP               Used to force a connection to a particular AP
  --number_template NUMBER_TEMPLATE
                        Start the station numbering with a particular number. Default is 0000
```

```
  --station_list STATION_LIST [STATION_LIST ...]
                        Optional: User defined station names, can be a comma or space separated list
  --batch_create        To enable batch create functionality
  --batch_quantity BATCH_QUANTITY
                        No of cx endpoints to batch-create
  --endp_a ENDP_A       --endp_a station list
  --endp_b ENDP_B       --upstream port
  --multi_conn_a MULTI_CONN_A
                        Modify multi connection endpoint-a for cx
  --multi_conn_b MULTI_CONN_B
                        Modify multi connection endpoint-b for cx
  --min_ip_port_a MIN_IP_PORT_A
                        Min ip port range for endp-a
  --min_ip_port_b MIN_IP_PORT_B
                        Min ip port range for endp-b
  --endp_a_increment ENDP_A_INCREMENT
                        End point - A port increment
  --endp_b_increment ENDP_B_INCREMENT
                        End point - B port increment
  --ip_port_increment_a IP_PORT_INCREMENT_A
                        Ip port increment for endp-a
  --ip_port_increment_b IP_PORT_INCREMENT_B
                        Ip port increment for endp-b

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

           Create stations to test connection and traffic on VAPs of varying security types (WEP, WPA, WPA2, WPA3, Open)
```

py-scripts/create_l4.py

```
usage: create_l4.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [-u UPSTREAM_PORT]
                    [--num_stations NUM_STATIONS] [--test_id TEST_ID] [-d]
                    [--log_level LOG_LEVEL]
                    [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                    [--proxy [PROXY]] [--debugging DEBUGGING [DEBUGGING ...]]
                    [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
                    [--radio RADIO] [--security SECURITY] [--ssid SSID]
                    [--passwd PASSWD] [--a_min A_MIN] [--b_min B_MIN]
                    [--mode MODE] [--ap AP] [--lf_user LF_USER]
                    [--lf_passwd LF_PASSWD]

---------------------------
Layer-4 Test Script - create_l4.py
---------------------------
Summary:
This script will create a user specified number of stations and layer-4 endpoints in a bytes-rd test scenario.
---------------------------
Generic command layout:

./create_l4.py
    --mgr <ip_address>
    --upstream_port eth1
    --radio wiphy0
    --num_stations 10
    --security {open|wep|wpa|wpa2|wpa3}
    --mode   1
        {"auto"   : "0",
        "a"      : "1",
        "b"      : "2",
        "g"      : "3",
        "abg"    : "4",
        "abgn"   : "5",
        "bgn"    : "6",
        "bg"     : "7",
        "abgnAC" : "8",
        "anAC"   : "9",
        "an"     : "10",
        "bgnAC"  : "11",
        "abgnAX" : "12",
        "bgnAX"  : "13",
    --ssid <ssid>
```

```
    --password <password>
    --a_min 1000
    --b_min 1000
    --ap "00:0e:8e:78:e1:76"
    --debug

EXAMPLE:
    ./create_l4.py --mgr <ip-address> --radio wiphy2 --num_stations 4 --upstream_port 1.1.eth1 --ssid <ssid> --passwd <passwd> --s

Tested on 02/13/2023:
        kernel version: 5.19.17+
        gui version: 5.4.6
        the layer-4 bytes-rd scenario was successfully created and tested on a ct523c sta-to-eth cross connection.



optional arguments:
  -h, --help            show this help message and exit
  --a_min A_MIN         --a_min bps rate minimum for side_a
  --b_min B_MIN         --b_min bps rate minimum for side_b
  --mode MODE           Used to force mode of stations
  --ap AP               Used to force a connection to a particular AP
  --lf_user LF_USER     --lf_user lanforge user name
  --lf_passwd LF_PASSWD
                        --lf_passwd lanforge password

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

            This script will create a specified number of stations and layer-4 endpoints as a bytes-rd scenario.
```

py-scripts/create_macvlan.py

```
usage: create_macvlan.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [--debug]
                         [--log_level LOG_LEVEL]
                         [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                         [--proxy [PROXY]] [--help_summary]
                         [--parent PARENT_PORT]
                         [--macvlan_ids MACVLAN_IDS [MACVLAN_IDS ...]]
                         [--cleanup]
                         [--dhcp | --ip IPV4_ADDRESSES [IPV4_ADDRESSES ...]]
                         [--netmask IPV4_NETMASKS [IPV4_NETMASKS ...]]
                         [--gateway IPV4_GATEWAYS [IPV4_GATEWAYS ...]]

NAME:       create_macvlan.py

PURPOSE:    Create and configure one or more MACVLAN ports using the specified parent port.

NOTES:      MACVLAN ports can only be created with a Ethernet, Bond, Redir, or 802.1Q VLAN port
            as the parent port.

            MACVLAN ports are different than 802.1Q VLAN ports.

            This script will optionally set IPv4 configuration (static or dynamic), if specified.
            The selected IPv4 configuration method will be applied to all ports created by this script.

EXAMPLE:    # Single MACVLAN with MACVLAN ID 10 on parent port '1.1.eth3'. No IPv4 configuration specified.
                ./create_macvlan.py                     --parent        1.1.eth3                    --macvlan_ids   10

            # Single MACVLAN with MACVLAN ID 10 on parent port '1.1.eth3'. DHCPv4 enabled.
                ./create_macvlan.py                     --parent        1.1.eth3                    --macvlan_ids   10

            # Single MACVLAN with MACVLAN ID 10 on parent port '1.1.eth3'. Static IPv4 configuration.
                ./create_macvlan.py                     --parent        1.1.eth3                    --macvlan_ids   10

            # Four MACVLANs with MACVLAN IDs 10, 20, 30, and 40 on parent port '1.1.eth3'. DHCPv4 enabled.
                ./create_macvlan.py                     --parent        1.1.eth3                    --macvlan_ids   10 20 30 40

            # Two MACVLANs with MACVLAN IDs 10 and 20 on parent port '1.1.eth3'. Static IPv4 configuration
                ./create_macvlan.py                     --parent        1.1.eth3                    --macvlan_ids   10 20
```

```
SCRIPT_CLASSIFICATION:
           Creation

SCRIPT_CATEGORIES:
           Functional

STATUS:    Functional

VERIFIED_ON:
           09-JUN-2023,
           GUI Version:  5.4.6
           Kernel Version: 5.19.17+

LICENSE:   Free to distribute and modify. LANforge systems must be licensed.
           Copyright 2024 Candela Technologies Inc.

INCLUDE_IN_README:
           False

optional arguments:
  -h, --help            show this help message and exit
  --parent PARENT_PORT, --parent_port PARENT_PORT, --macvlan_parent PARENT_PORT
                        Parent port used by created MACVLAN port(s)
  --macvlan_ids MACVLAN_IDS [MACVLAN_IDS ...]
                        MACVLAN ID(s) used in creation. One MACVLAN port is created per ID. For static IP configuration, the numbe
  --cleanup             Clean up any created ports before exiting.
  --dhcp, --dhcpv4, --use_dhcp
                        Enable DHCPv4 on created ports
  --ip IPV4_ADDRESSES [IPV4_ADDRESSES ...], --ips IPV4_ADDRESSES [IPV4_ADDRESSES ...], --ipv4_address IPV4_ADDRESSES [IPV4_ADDRESS
                        List of static IPv4 addresses. The number of IPv4 addresses specified must match the number of ports speci
  --netmask IPV4_NETMASKS [IPV4_NETMASKS ...], --netmasks IPV4_NETMASKS [IPV4_NETMASKS ...], --ipv4_netmask IPV4_NETMASKS [IPV4_NE
                        IPv4 subnet mask to apply to all created MACVLAN ports when static IPv4 configuration requested
  --gateway IPV4_GATEWAYS [IPV4_GATEWAYS ...], --gateways IPV4_GATEWAYS [IPV4_GATEWAYS ...], --ipv4_gateway IPV4_GATEWAYS [IPV4_GA
                        IPv4 gateway to apply to all created MACVLAN ports when static IPv4 configuration requested

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_bare_argparse found in /lanforge-scripts/py-json/LANfo
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --debug, -d           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80 or https://user:pass@proxy.localnet:3128
  --help_summary        Show summary of what this script does

           Creates MACVLAN endpoints.
```

py-scripts/create_qvlan.py

```
usage: create_qvlan.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [--debug]
                       [--log_level LOG_LEVEL]
                       [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                       [--proxy [PROXY]] [--help_summary]
                       [--parent PARENT_PORT]
                       [--qvlan_ids QVLAN_IDS [QVLAN_IDS ...]]
                       [--dhcp | --ip IPV4_ADDRESSES [IPV4_ADDRESSES ...]]
                       [--netmask IPV4_NETMASKS [IPV4_NETMASKS ...]]
                       [--gateway IPV4_GATEWAYS [IPV4_GATEWAYS ...]]

NAME:       create_qvlan.py

PURPOSE:    Create and configure one or more QVLAN port using the specified parent port.

NOTES:      This script will optionally set IPv4 configuration (static or dynamic), if specified.
            The selected IPv4 configuration method will be applied to all created QVLAN ports
            created by this script.

EXAMPLE:    # Single QVLAN with QVLAN ID 10 on parent port '1.1.eth3'. No IPv4 configuration specified.
                ./create_qvlan.py                   --parent        1.1.eth3                    --qvlan_ids     10

            # Single QVLAN with QVLAN ID 10 on parent port '1.1.eth3'. DHCPv4 enabled.
                ./create_qvlan.py                   --parent        1.1.eth3                    --qvlan_ids     10

            # Single QVLAN with QVLAN ID 10 on parent port '1.1.eth3'. Static IPv4 configuration.
                ./create_qvlan.py                   --parent        1.1.eth3                    --qvlan_ids     10

            # Four QVLANs with QVLAN IDs 10, 20, 30, and 40 on parent port '1.1.eth3'. DHCPv4 enabled.
                ./create_qvlan.py                   --parent        1.1.eth3                    --qvlan_ids     10 20 30 40

            # Two QVLANs with QVLAN IDs 10 and 20 on parent port '1.1.eth3'. Static IPv4 configuration
                ./create_qvlan.py                   --parent        1.1.eth3                    --qvlan_ids     10 20

LICENSE:    Free to distribute and modify. LANforge systems must be licensed.
            Copyright 2024 Candela Technologies Inc.

optional arguments:
  -h, --help            show this help message and exit
  --parent PARENT_PORT, --parent_port PARENT_PORT, --qvlan_parent PARENT_PORT
                        Parent port used by created QVLAN port(s)
  --qvlan_ids QVLAN_IDS [QVLAN_IDS ...]
                        QVLAN ID(s) used in creation. One QVLAN port is created per ID. For static IP configuration, the number of
  --dhcp, --dhcpv4, --use_dhcp
                        Enable DHCPv4 on created QVLAN ports
  --ip IPV4_ADDRESSES [IPV4_ADDRESSES ...], --ips IPV4_ADDRESSES [IPV4_ADDRESSES ...], --ipv4_address IPV4_ADDRESSES [IPV4_ADDRESS
                        List of static IPv4 addresses. The number of IPv4 addresses specified must match the number of QVLAN ports
  --netmask IPV4_NETMASKS [IPV4_NETMASKS ...], --netmasks IPV4_NETMASKS [IPV4_NETMASKS ...], --ipv4_netmask IPV4_NETMASKS [IPV4_NE
                        IPv4 subnet mask to apply to all created QVLAN ports when static IPv4 configuration requested
  --gateway IPV4_GATEWAYS [IPV4_GATEWAYS ...], --gateways IPV4_GATEWAYS [IPV4_GATEWAYS ...], --ipv4_gateway IPV4_GATEWAYS [IPV4_GA
                        IPv4 gateway to apply to all created QVLAN ports when static IPv4 configuration requested

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_bare_argparse found in /lanforge-scripts/py-json/LANfo
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --debug, -d           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
```

```
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80 or https://user:pass@proxy.localnet:3128
  --help_summary        Show summary of what this script does
```

py-scripts/create_station_from_df.py

```
usage: create_station_from_df.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                                 [-u UPSTREAM_PORT]
                                 [--num_stations NUM_STATIONS]
                                 [--test_id TEST_ID] [-d]
                                 [--log_level LOG_LEVEL]
                                 [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                 [--proxy [PROXY]]
                                 [--debugging DEBUGGING [DEBUGGING ...]]
                                 [--debug_log DEBUG_LOG] [--no_cleanup]
                                 [--help_summary] [--radio RADIO]
                                 [--security SECURITY] [--ssid SSID]
                                 [--passwd PASSWD] [--df DF]

       create_station.py
       --------------------
       Command example:
       ./create_station_from_df.py
           --upstream_port eth1
           --df df.csv
           --security open
           --ssid netgear
           --passwd BLANK
           --debug


optional arguments:
  -h, --help            show this help message and exit

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

required arguments:
  --df DF               Which file do you want to build stations off of?

        Create stations
```

py-scripts/create_station.py

```
usage: create_station.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                         [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                         [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                         [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                         [--proxy [PROXY]]
                         [--debugging DEBUGGING [DEBUGGING ...]]
                         [--debug_log DEBUG_LOG] [--no_cleanup]
                         [--help_summary] [--radio RADIO]
                         [--security SECURITY] [--ssid SSID] [--passwd PASSWD]
                         [--start_id START_ID] [--prefix PREFIX]
                         [--create_admin_down] [--bssid BSSID] [--mode MODE]
                         [--station_flags STATION_FLAGS]
                         [--mac_pattern MAC_PATTERN]
                         [--radio_antenna RADIO_ANTENNA]
                         [--radio_channel RADIO_CHANNEL]
                         [--radio_tx_power RADIO_TX_POWER]
                         [--country_code COUNTRY_CODE]
                         [--eap_method EAP_METHOD]
                         [--eap_identity EAP_IDENTITY]
                         [--eap_anonymous_identity EAP_ANONYMOUS_IDENTITY]
                         [--eap_password EAP_PASSWORD]
                         [--eap_phase1 EAP_PHASE1] [--eap_phase2 EAP_PHASE2]
                         [--pk_passwd PK_PASSWD] [--ca_cert CA_CERT]
                         [--private_key PRIVATE_KEY] [--key_mgmt KEY_MGMT]
                         [--pairwise_cipher PAIRWISE_CIPHER]
                         [--groupwise_cipher GROUPWISE_CIPHER]
                         [--no_pre_cleanup] [--cleanup]
```

```
                        [--custom_wifi_cmd CUSTOM_WIFI_CMD]

NAME:       create_station.py

PURPOSE:    Create and configure one or more of WiFi stations ports using the
            specified parent radio.

NOTES:      This script is intended to only create and configure stations. See other scripts
            like 'test_l3.py' to create and run tests.

            By default, the script will also attempt to connect the WiFi stations as configured
            unless the '--create_admin_down' argument is specified.

            --mode <mode_num>
                Set the station WiFi mode (e.g. to configure a 802.11be radio as 802.11ax)
                See the 'add_sta' command's mode option in the CLI documentation for
                available mode settings. Link here: http://www.candelatech.com/lfcli_ug.php#add_sta

            --station_flags  <station_flags>
                Comma-separated list of flags to configure the station with (e.g. 'ht160_enable,disable_sgi'
                to enable 160MHz channel usage and disable 802.11ac short guard interval (SGI), respectively).
                Note that other options like '--security' configure authentication-based station flags.
                See the 'add_sta' command's 'flags' option in the CLI documentation for available options.
                Link here: http://www.candelatech.com/lfcli_ug.php#add_sta

            --country_code 840
                United States   :   840     |       Dominican Rep   :   214     |       Japan (JE2)     :   397     |       Portugal
                Albania         :   8       |       Ecuador         :   218     |       Jordan          :   400     |       Pueto Ri
                Algeria         :   12      |       Egypt           :   818     |       Kazakhstan      :   398     |       Qatar
                Argentina       :   32      |       El Salvador     :   222     |       North Korea     :   408     |       Romania
                Bangladesh      :   50      |       Estonia         :   233     |       South Korea     :   410     |       Russia
                Armenia         :   51      |       Finland         :   246     |       South Korea     :   411     |       Saudi Ar
                Australia       :   36      |       France          :   250     |       Kuwait          :   414     |       Singapor
                Austria         :   40      |       Georgia         :   268     |       Latvia          :   428     |       Slovak R
                Azerbaijan      :   31      |       Germany         :   276     |       Lebanon         :   422     |       Slovenia
                Bahrain         :   48      |       Greece          :   300     |       Liechtenstein   :   438     |       South Af
                Barbados        :   52      |       Guatemala       :   320     |       Lithuania       :   440     |       Spain
                Belarus         :   112     |       Haiti           :   332     |       Luxembourg      :   442     |       Sweden
                Belgium         :   56      |       Honduras        :   340     |       Macau           :   446     |       Switzerl
                Belize          :   84      |       Hong Kong       :   344     |       Macedonia       :   807     |       Syria
                Bolivia         :   68      |       Hungary         :   348     |       Malaysia        :   458     |       Taiwan
                BiH             :   70      |       Iceland         :   352     |       Mexico          :   484     |       Thailand
                Brazil          :   76      |       India           :   356     |       Monaco          :   492     |       Trinidad
                Brunei          :   96      |       Indonesia       :   360     |       Morocco         :   504     |       Tunisia
                Bulgaria        :   100     |       Iran            :   364     |       Netherlands     :   528     |       Turkey
                Canada          :   124     |       Ireland         :   372     |       Aruba           :   533     |       U.A.E.
                Chile           :   152     |       Israel          :   376     |       New Zealand     :   554     |       Ukraine
                China           :   156     |       Italy           :   380     |       Norway          :   578     |       United K
                Colombia        :   170     |       Jamaica         :   388     |       Oman            :   512     |       Uruguay
                Costa Rica      :   188     |       Japan           :   392     |       Pakistan        :   586     |       Uzbekist
                Croatia         :   191     |       Japan (JP1)     :   393     |       Panama          :   591     |       Venezuel
                Cyprus          :   196     |       Japan (JP0)     :   394     |       Peru            :   604     |       Vietnam
                Czech Rep       :   203     |       Japan (JP1-1)   :   395     |       Philippines     :   608     |       Yemen
                Denmark         :   208     |       Japan (JE1)     :   396     |       Poland          :   616     |       Zimbabwe

            --no_pre_cleanup
                Disables station cleanup before creation of stations. Default behavior will remove any existing stations.

            --cleanup
                Add this flag to clean up stations after creation

            --eap_method <eap_method>
                EAP method used by station in authentication.
                See the 'set_wifi_extra' command's 'eap' option in the CLI documentation for available options.
                Link here: http://www.candelatech.com/lfcli_ug.php#set_wifi_extra

            --key_mgmt <protocol>
                Key management protocol used by the station in authentication.

            --pairwise_cipher <cipher>
                Pairwise cipher used by station in authentication.
                See the 'set_wifi_extra' command's 'pairwise' option in the CLI documentation for available options.
                Link here: http://www.candelatech.com/lfcli_ug.php#set_wifi_extra

            --groupwise_cipher <cipher>
                Groupwise cipher used by station in authentication.
                See the 'set_wifi_extra' command's 'groupwise' option in the CLI documentation for available options.
                Link here: http://www.candelatech.com/lfcli_ug.php#set_wifi_extra

            --eap_identity <eap_identity>
                EAP identity (i.e. username) used by the station in authentication.

            --eap_password <eap_password>
                EAP password used by the station in authentication.

            --pk_passwd <password>
                Private key password used by the station in authentication. Required for TLS-based authentication.

            --ca_cert <path_to_certificate>
                Path to Certificate authority certificate used by the station in authentication.
                Required for TLS-based authentication.
                Note this is the path on the LANforge system where this station will be created.

            --private_key <path_to_private_key>
                Path to private key used by the station in authentication. Required for TLS-based authentication.
                Note this is the path on the LANforge system where this station will be created.

EXAMPLE:    # Create a single station
               ./create_station.py                  --mgr       <lanforge ip>                   --radio     1.1.wiphy1

            # Create multiple stations
               ./create_station.py                  --mgr          <lanforge ip>                    --radio         1.1.wiphy1

            # Create a multiple stations with specific numbering scheme
            # In this example, create five stations with names of the format: "sta1000", "sta1001", "sta1002", etc.
               ./create_station.py                  --mgr          <lanforge ip>                    --radio         1.1.wiphy1
```

```
            # Create a station, configuring radio settings like antenna, channel, etc.
            # In this example, configure radio to use antennas (2x2 station) and channel 6
               ./create_station.py                  --mgr          <lanforge ip>                  --radio        1.1.wiphy1

            # Create a station, configuring the station to be a specific WiFi mode
            # (e.g. configuring an 802.11ax-capable radio to create an 802.11ac station)
            # See the 'add_sta' command's mode option in the CLI documentation for
            # available mode settings. Link here: http://www.candelatech.com/lfcli_ug.php#add_sta
               ./create_station.py                  --mgr       <lanforge ip>                   --radio     1.1.wiphy1

            # Create a station, configuring specific station flags
            # In this example, enable 160MHz channels and disable 802.11ac short guard interval (SGI).
               ./create_station.py                  --mgr          <lanforge ip>                  --radio        1.1.wiphy1

            # Create a station using TLS-based enterprise authentication
            # Note that paths are paths on the LANforge system where the station will be created.
               ./create_station.py                  --mgr              <lanforge ip>                  --radio            1.1

            # Create a station using TTLS-based enterprise authentication
               ./create_station.py                  --mgr              <lanforge ip>                  --radio            1.1

            #    Create station specifying a custom 'wpa_supplicant' config command
            #    In this example, specify a background scanning 'wpa_supplicant' command, useful for roaming.
            #    Here, the background scan is configured to a threshold of -65 dBm RSSI with a short and long interval of 50 and 3
            #    See 'man wpa_supplicant.conf' for more information.
                  ./create_station.py                       --mgr              <lanforge ip>                       --radio

SCRIPT_CLASSIFICATION:
            Creation

SCRIPT_CATEGORIES:
            Functional

STATUS:     Functional

VERIFIED_ON:
            9-JUN-2023,
            GUI Version:  5.4.6
            Kernel Version: 5.19.17+

LICENSE:    Free to distribute and modify. LANforge systems must be licensed.
            Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README:
            False

optional arguments:
  -h, --help            show this help message and exit

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

required arguments:
  --start_id START_ID   Specify the station starting id
                         e.g: --start_id <value> default 0

Optional arguments:
  --prefix PREFIX       Station prefix. Default: 'sta'
  --create_admin_down   Create ports in admin down state.
  --bssid BSSID         AP BSSID. For example, "00:00:00:00:00:00".
  --mode MODE           Mode for your station (as a number)
  --station_flags STATION_FLAGS, --station_flag STATION_FLAGS
                        station flags to add. eg: --station_flags ht40_disable
  --mac_pattern MAC_PATTERN
                        MAC randomization pattern for created stations. In full MAC address pattern, the '*' indicates randomizabl
  --radio_antenna RADIO_ANTENNA
                        Number of spatial streams:
                         default = -1
                         0 Diversity (All)
                         1 Fixed-A (1x1)
                         4 AB (2x2)
                         7 ABC (3x3)
                         8 ABCD (4x4)
```

```
                        9 (8x8)
  --radio_channel RADIO_CHANNEL
                       Radio Channel:
                        default: AUTO
                        e.g:   --radio_channel 6 (2.4G)
                        --radio_channel 36 (5G)
  --radio_tx_power RADIO_TX_POWER
                       Radio tx-power
                        default: AUTO system defaults
  --country_code COUNTRY_CODE
                       Radio Country Code:
                       e.g:  --country_code 840
  --eap_method EAP_METHOD
                       Enter EAP method e.g: TLS
  --eap_identity EAP_IDENTITY, --radius_identity EAP_IDENTITY
                       This is synonymous with the RADIUS username.
  --eap_anonymous_identity EAP_ANONYMOUS_IDENTITY
  --eap_password EAP_PASSWORD, --radius_passwd EAP_PASSWORD
                       This is synonymous with the RADIUS user's password.
  --eap_phase1 EAP_PHASE1
                       EAP Phase 1 (outer authentication, i.e. TLS tunnel) parameters.
                       For example, "peapver=0" or "peapver=1 peaplabel=1".
                       Some WPA Enterprise setups may require "auth=MSCHAPV2"
  --eap_phase2 EAP_PHASE2
                       EAP Phase 2 (inner authentication) parameters.
                       For example, "autheap=MSCHAPV2 autheap=MD5" for EAP-TTLS.
  --pk_passwd PK_PASSWD
                       Enter the private key password
  --ca_cert CA_CERT     Enter path for certificate e.g: /home/lanforge/ca.pem
  --private_key PRIVATE_KEY
                       Enter private key path e.g: /home/lanforge/client.p12
  --key_mgmt KEY_MGMT   Authentication key management. Combinations are supported.
  --pairwise_cipher PAIRWISE_CIPHER
                       Pairwise Ciphers
                       DEFAULT
                       CCMP
                       TKIP
                       NONE
                       CCMP-TKIP
                       CCMP-256
                       GCMP
                       GCMP-256
                       CCMP/GCMP-256
  --groupwise_cipher GROUPWISE_CIPHER
                       Groupwise Ciphers
                       DEFAULT
                       CCMP
                       TKIP
                       WEP104
                       WEP40
                       GTK_NOT_USED
                       GCMP-256
                       CCMP-256
                       GCMP/CCMP-256
                       ALL
  --no_pre_cleanup      Add this flag to stop cleaning up before station creation
  --cleanup             Add this flag to clean up stations after creation
  --custom_wifi_cmd CUSTOM_WIFI_CMD
                       Mention the custom wifi command.

        Create stations
```

py-scripts/create_vap.py

```
usage: create_vap.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [-u UPSTREAM_PORT]
                     [--num_stations NUM_STATIONS] [--test_id TEST_ID] [-d]
                     [--log_level LOG_LEVEL]
                     [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                     [--proxy [PROXY]] [--debugging DEBUGGING [DEBUGGING ...]]
                     [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
                     [--radio RADIO] [--security SECURITY] [--ssid SSID]
                     [--passwd PASSWD] [--num_vaps NUM_VAPS]
                     [--vap_flag VAP_FLAG] [--mac MAC] [--mode MODE]
                     [--channel CHANNEL] [--country_code COUNTRY_CODE]
                     [--resource RESOURCE] [--start_id START_ID]
                     [--vap_suffix VAP_SUFFIX]

--------------------
NAME: create_vap.py

PURPOSE:
    create_vap.py will create a variable number of VAPs.

EXAMPLE:
    ./create_vap.py
    --lfmgr <lanforge ip>
    --port <lanforge port 8080>
    --upstream_port eth1
    --radio wiphy0
    --num_vaps 3
    --security open
    --ssid netgear
    --passwd BLANK
    --debug

NOTES:

    Tested on 03/21/2023:
         kernel version: 5.19.17+
         gui version: 5.4.6



optional arguments:
  -h, --help            show this help message and exit
```

```
arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

optional arguments:
  --num_vaps NUM_VAPS   Number of VAPs to Create
  --vap_flag VAP_FLAG   VAP flags to add
  --mac MAC             Custom mac address
  --mode MODE
  --channel CHANNEL
  --country_code COUNTRY_CODE
  --resource RESOURCE
  --start_id START_ID
  --vap_suffix VAP_SUFFIX
                        The numeric suffix, like the 005 in vap005

        Create VAPs
```

py-scripts/create_vr.py

```
usage: ./create_vr.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [--debug]
                      [--log_level LOG_LEVEL]
                      [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                      [--proxy [PROXY]] [--help_summary] [--vr_name VR_NAME]
                      [--ports PORTS] [--services SERVICES]

--------------------
NAME: create_vr.py

PURPOSE:
    create_vr.py will create a variable number of virtual routers.

EXAMPLE:
    ./create_vr.py --vr_name 1.vr0 --ports 1.br0,1.rdd0a --services 1.br0=dhcp,nat --services 1.vr0=radvd
    ./create_vr.py --vr_name 2.vr0 --ports 2.br0,2.vap2 --services

NOTES:

    Tested on 03/21/2023:
         kernel version: 5.19.17+
         gui version: 5.4.6

optional arguments:
  -h, --help            show this help message and exit

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_bare_argparse found in /lanforge-scripts/py-json/LANfo
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --debug, -d           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80 or https://user:pass@proxy.localnet:3128
  --help_summary        Show summary of what this script does

required arguments:
  --vr_name VR_NAME, --vr_names VR_NAME
                        EID of virtual router, like 1.2.vr0

optional arguments:
  --ports PORTS         Comma separated list of ports to add to virtual router
  --services SERVICES   Add router services to a port, "br0=nat,dhcp"
```

py-scripts/csv_convert.py

```
usage: csv_convert.py [-h] [-i INFILE] [-I INFILE2] [-o OUTFILE]
                      [--help_summary]

csv_convert.py:
    converts the candela brief csv and/or more complete csv into the data for specific customer.
    Both csv files need to be passed in order to have beacon rssi and phy rates since neither
    csv file contains all of that data.
```

```
Example:
    ./csv_convert.py -i ~/dataplane-2022-02-08-12-18-45/text-csv-2.csv -I ~/dataplane-2022-02-08-12-18-45/text-csv-0.csv


optional arguments:
  -h, --help            show this help message and exit
  -i INFILE, --infile INFILE
                        input file of csv data
  -I INFILE2, --infile2 INFILE2
                        secondary input file of csv data
  -o OUTFILE, --outfile OUTFILE
                        output file in .csv format
  --help_summary        Show summary of what this script does

 Useful Information:
```

py-scripts/csv_processor.py

```
usage: csv_processor.py [-h] [-i INFILE] [--debug DEBUG] [--help_summary]

csv_processor.py:



optional arguments:
  -h, --help            show this help message and exit
  -i INFILE, --infile INFILE
                        file of csv data
  --debug DEBUG         --debug:  Enable debugging
  --help_summary        Show summary of what this script does

 This script is an simple example on how to process data from a csv file.
```

py-scripts/cv_manager.py

```
usage: cv_manager.py [-h] [--scenario SCENARIO] [--debug] [--mgr MGR]
                     [--help_summary]

This is a simple driver script to load a CV Scenario

optional arguments:
  -h, --help           show this help message and exit
  --scenario SCENARIO  Scenario you wish to build
  --debug              Enable debugging
  --mgr MGR
  --help_summary       Show summary of what this script does
```

py-scripts/lf_add_profile.py

```
usage: ./lf_add_profile.py [-h] [--host MGR] [--mgr_port MGR_PORT]
                           [--lf_user LF_USER] [--lf_passwd LF_PASSWD]
                           [--alias_prefix ALIAS_PREFIX] [--antenna ANTENNA]
                           [--bandwidth BANDWIDTH] [--eap_id EAP_ID]
                           [--flags_mask FLAGS_MASK] [--freq FREQ]
                           [--instance_count INSTANCE_COUNT]
                           [--mac_pattern MAC_PATTERN] [--name NAME]
                           [--passwd PASSWD] [--profile_flags PROFILE_FLAGS]
                           [--profile_flags_hex PROFILE_FLAGS_HEX] [--pf PF]
                           [--profile_type PROFILE_TYPE] [--ssid SSID]
                           [--vid VID] [--wifi_mode WIFI_MODE] [--dut DUT]
                           [--text TEXT] [--log_level LOG_LEVEL]
                           [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                           [--debug] [--help_summary]

           adds a chamberview profile

           add_profile Routed-AP-QA Routed-AP 0 4 1 0 vap hello123 4105

           profile flags
           4105 = 0x1009

           DHCP_SERVER = 0x1           # This should provide DHCP server.
           WPA2        = 0x8
           ENABLE_POWERSAVE = 0x1000   # Enable power-save when creating stations.

           pass in --profile_flags 'DHCP_SERVER,WPA2,ENABLE_POWERSAVE'

    Example:
        Command line:
        see http://www.candelatech.com/lfcli_ug.php#add_profile for the profile flags:
        ./lf_add_profile.py --mgr 192.168.0.104 --mgr_port 8080 --lf_user lanforge --lf_passwd lanforge --antenna 4            --i

        Once the profile is created a chamberview scenario needs to be created based off that profile:
        ./create_chamberview.py --lfmgr 192.168.0.104 --port 8080 --delete_scenario --create_scenario QA13-2        --raw_line 'pr
        The --raw_line are determined by applying the profile above

    NOTES:
        Tested on 02/20/2023:
        kernel version: 5.19.17+
        gui version: 5.4.6



optional arguments:
  -h, --help            show this help message and exit
  --host MGR, --mgr MGR
                        specify the GUI to connect to
  --mgr_port MGR_PORT   specify the GUI to connect to, default 8080
  --lf_user LF_USER     lanforge user name
  --lf_passwd LF_PASSWD
                        lanforge password
  --alias_prefix ALIAS_PREFIX
                        (add profile) alias_prefix Port alias prefix, aka hostname prefix.
  --antenna ANTENNA     (add profile) Antenna count for this profile.
```

```
--bandwidth BANDWIDTH
                      (add profile) 0 (auto), 20, 40, 80 or 160
--eap_id EAP_ID       (add profile) EAP Identifier
--flags_mask FLAGS_MASK
                      (add profile) Specify what flags to set.
--freq FREQ           (add profile)WiFi frequency to be used, 0 means default.
--instance_count INSTANCE_COUNT
                      (add profile) Number of devices (stations, vdevs, etc)
--mac_pattern MAC_PATTERN
                      (add profile) Optional MAC-Address pattern, for instance:  xx:xx:xx:*:*:xx
--name NAME           (add profile) Profile Name. [R]
--passwd PASSWD       (add profile) WiFi SSID to be used, [BLANK] means any.
--profile_flags PROFILE_FLAGS
                      pass in flags as a decimal value takes presidence over hex and --pf []
--profile_flags_hex PROFILE_FLAGS_HEX
                      pass in flags as a hex value
--pf PF
                          (add profile)
                          Profile flags entered as a list
                          Flags for this profilelanforge_api AddProfileProfileFlags'
                          enter the flags as a list 0x1009 is:
                              DHCP_SERVER = 0x1           # This should provide DHCP server.
                              SKIP_DHCP_ROAM = 0x10       # Ask station to not re-do DHCP on roam.
                              NAT = 0x100                 # Enable NAT if this object is in a virtual router
                              ENABLE_POWERSAVE = 0x1000   # Enable power-save when creating stations.

                              pass in --profile_flags 'DHCP_SERVER,SKIP_DHCP_ROAM,NAT,ENABLE_POWERSAVE'

                          flags:
                                  p_11r = 0x40                  # Use 802.11r roaming setup.
                                  ALLOW_11W = 0x800             # Set 11w (MFP/PMF) to optional.
                                  BSS_TRANS = 0x400             # Enable BSS Transition logic
                                  DHCP_SERVER = 0x1             # This should provide DHCP server.
                                  EAP_PEAP = 0x200              # Enable EAP-PEAP
                                  EAP_TTLS = 0x80               # Use 802.1x EAP-TTLS
                                  ENABLE_POWERSAVE = 0x1000     # Enable power-save when creating stations.
                                  NAT = 0x100                   # Enable NAT if this object is in a virtual router
                                  SKIP_DHCP_ROAM = 0x10         # Ask station to not re-do DHCP on roam.
                                  WEP = 0x2                     # Use WEP encryption
                                  WPA = 0x4                     # Use WPA encryption
                                  WPA2 = 0x8                    # Use WPA2 encryption
                                  WPA3 = 0x20                   # Use WPA3 encryption

--profile_type PROFILE_TYPE

                              (add profile)
                              Profile type: [W]
                              Bridged_AP: briged-AP
                              Monitor: monitor
                              Peer: peer
                              RDD: rdd
                              Routed_AP: routed-AP
                              STA: STA-AC
                              STA: STA-AUTO
                              STA: STA-AX
                              STA: STA-abg
                              STA: STA-n
                              Uplink: uplink-nat
                              Upstream: upstrream
                              Upstream: upstream-dhcp
                              as_is: as-is
                              NA

--ssid SSID           (add profile) WiFi SSID to be used, [BLANK] means any.
--vid VID             (add profile) Vlan-ID (only valid for vlan profiles).
--wifi_mode WIFI_MODE
                      (add profile) WiFi Mode for this profile.
                              p_802_11a = "802.11a"         # 802.11a
                              AUTO = "AUTO"                 # 802.11g
                              aAX = "aAX"                   # 802.11a-AX (6E disables /n and /ac)
                              abg = "abg"                   # 802.11abg
                              abgn = "abgn"                 # 802.11abgn
                              abgnAC = "abgnAC"             # 802.11abgn-AC
                              abgnAX = "abgnAX"             # 802.11abgn-AX
                              an = "an"                     # 802.11an
                              anAC = "anAC"                 # 802.11an-AC
                              anAX = "anAX"                 # 802.11an-AX
                              as_is = "as_is"               # Make no changes to current configuration
                              b = "b"                       # 802.11b
                              bg = "bg"                     # 802.11bg
                              bgn = "bgn"                   # 802.11bgn
                              bgnAC = "bgnAC"               # 802.11bgn-AC
                              bgnAX = "bgnAX"               # 802.11bgn-AX
                              bond = "bond"                 # Bonded pair of Ethernet ports.
                              bridged_ap = "bridged_ap"     # AP device in bridged mode. The EIDs may specify radio and bridged p
                              client = "client"             # Client-side non-WiFi device (Ethernet port, for instance).
                              g = "g"                       # 802.11g
                              mobile_sta = "mobile_sta"     # Mobile station device. Expects to connect to DUT AP(s) and upstream
                              monitor = "monitor"           # Monitor device/sniffer. The EIDs may specify which radios to use.
                              peer = "peer"                 # Edge device, client or server (Ethernet port, for instance).
                              rdd = "rdd"                   # Pair of redirect devices, typically associated with VR to act as tr
                              routed_ap = "routed_ap"       # AP in routed mode. The EIDs may specify radio and upstream port.
                              sta = "sta"                   # Station device, most likely non mobile. The EIDs may specify radio(
                              uplink = "uplink"             # Uplink towards rest of network (can go in virtual router and do NAT
                              upstream = "upstream"         # Upstream server device. The EIDs may specify which ports to use.
                              vlan = "vlan"                 # 802.1q VLAN. Specify VID with the 'freq' option.

--dut DUT             (add profile notes) Profile Name. [R]
--text TEXT           (add profile notes) list of lines of text
                                                            [BLANK] will erase all,
                                                            any other text will be appended to existing text.
                                                            must be entered line by line

--log_level LOG_LEVEL
                      Set logging level: debug | info | warning | error | critical
--lf_logger_config_json LF_LOGGER_CONFIG_JSON
                      --lf_logger_config_json <json file> , json configuration of logger
```

```
  --debug               Legacy debug flag
  --help_summary        shows summary of the script
```

py-scripts/lf_ap_auto_test.py

```
usage: lf_ap_auto_test.py [-h] [-m MGR] [-o PORT] [--lf_user LF_USER]
                          [--lf_password LF_PASSWORD] [-i INSTANCE_NAME]
                          [-c CONFIG_NAME] [-r] [--load_old_cfg]
                          [--enable ENABLE] [--disable DISABLE]
                          [--set SET SET] [--raw_line RAW_LINE]
                          [--raw_lines_file RAW_LINES_FILE]
                          [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                          [-u UPSTREAM] [--max_stations_2 MAX_STATIONS_2]
                          [--max_stations_5 MAX_STATIONS_5]
                          [--max_stations_dual MAX_STATIONS_DUAL]
                          [--dut5_0 DUT5_0] [--dut2_0 DUT2_0]
                          [--radio2 RADIO2] [--radio5 RADIO5]
                          [--verbosity VERBOSITY]
                          [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                          [--lf_report_dir LF_REPORT_DIR] [--debug]
                          [--log_level LOG_LEVEL]
                          [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                          [--help_summary]

   Open this file in an editor and read the top notes for more details.

   Example:
   ./lf_ap_auto_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge \
     --instance_name ap-auto-instance --config_name test_con --upstream 1.1.eth2 \
     --dut5_0 'linksys-8450 Default-SSID-5gl c4:41:1e:f5:3f:25 (2)' \
     --dut2_0 'linksys-8450 Default-SSID-2g c4:41:1e:f5:3f:24 (1)' \
     --max_stations_2 100 --max_stations_5 100 --max_stations_dual 200 \
     --radio2 1.1.wiphy0 --radio2 1.1.wiphy2 \
     --radio5 1.1.wiphy1 --radio5 1.1.wiphy3 --radio5 1.1.wiphy4 \
     --radio5 1.1.wiphy5 --radio5 1.1.wiphy6 --radio5 1.1.wiphy7 \
     --set 'Basic Client Connectivity' 1 --set 'Multi Band Performance' 1 \
     --set 'Skip 2.4Ghz Tests' 1 --set 'Skip 5Ghz Tests' 1 \
     --set 'Throughput vs Pkt Size' 0 --set 'Capacity' 0 --set 'Stability' 0 --set 'Band-Steering' 0 \
     --set 'Multi-Station Throughput vs Pkt Size' 0 --set 'Long-Term' 0 \
     --test_rig Testbed-01 --test_tag ATH10K --pull_report


optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the capacity test?  Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).  See example raw text config for possible options.  M
  --disable DISABLE     Specify options to disable (set value to 0).  See example raw text config for possible options.  May be sp
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for instance:  testbed-01
  -u UPSTREAM, --upstream UPSTREAM
                        Upstream port for wifi capacity test ex. 1.1.eth1
  --max_stations_2 MAX_STATIONS_2
                        Specify maximum 2.4Ghz stations
  --max_stations_5 MAX_STATIONS_5
                        Specify maximum 5Ghz stations
  --max_stations_dual MAX_STATIONS_DUAL
                        Specify maximum stations for dual-band tests
  --dut5_0 DUT5_0       Specify 5Ghz DUT entry.  Syntax is somewhat tricky:  DUT-name SSID BSID (bssid-idx), example: linksys-8450
  --dut2_0 DUT2_0       Specify 5Ghz DUT entry.  Syntax is somewhat tricky:  DUT-name SSID BSID (bssid-idx), example: linksys-8450
  --radio2 RADIO2       Specify 2.4Ghz radio.  May be specified multiple times.
  --radio5 RADIO5       Specify 5Ghz radio.  May be specified multiple times.
  --verbosity VERBOSITY
                        Specify verbosity of the report values 1 - 11 default 5
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir <where to pull reports to>  default '' put where dataplane script run from
  --lf_report_dir LF_REPORT_DIR
                        --lf_report_dir <where to pull reports from>  default '' put where dataplane script run from
  --debug               Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --help_summary        Show summary of what this script does
```

py-scripts/lf_atten_mod_test.py

```
usage: lf_atten_mod_test.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                            [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                            [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                            [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                            [--proxy [PROXY]]
                            [--debugging DEBUGGING [DEBUGGING ...]]
                            [--debug_log DEBUG_LOG] [--no_cleanup]
                            [--help_summary] [--radio RADIO]
                            [--security SECURITY] [--ssid SSID]
                            [--passwd PASSWD] [--atten_serno ATTEN_SERNO]
                            [--atten_idx ATTEN_IDX] [--atten_val ATTEN_VAL]

NAME:       lf_atten_mod_test.py

PURPOSE:    lf_atten_mod_test.py is used to modify and/or read the LANforge Attenuator settings.

EXAMPLE:    Set channel four (zero-indexed) of all attenuators on LANforge system '192.168.200.12'
```

```
            to attenuation value 220 ddB (22 dB).
            Command: './lf_atten_mod_test.py --mgr 192.168.200.12 --atten_serno all --atten_idx 3 --atten_val 220'

            Set channel all channels of attenuator 2324 on LANforge system '192.168.200.12'
            to attenuation value 0 ddB (0.0 dB).
            Command: './lf_atten_mod_test.py --mgr 192.168.200.12 --atten_serno 2324 --atten_idx all --atten_val 0'

optional arguments:
  -h, --help            show this help message and exit
  --atten_serno ATTEN_SERNO
                        Serial number for requested attenuator, or 'all'
  --atten_idx ATTEN_IDX
                        Attenuator index eg. For module 1 = 0, module 2 = 1, or 'all'
  --atten_val ATTEN_VAL
                        Requested attenuation in 1/10ths of dB (ddB).

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key
```

py-scripts/lf_base_interop_profile.py

```
usage: ./lf_base_interop_profile.py [-h] [--debug] [--host HOST] [--ssid SSID]
                                    [--crypt CRYPT] [--passwd PASSWD]
                                    [--ssid_2g SSID_2G]
                                    [--encryption_2g ENCRYPTION_2G]
                                    [--passwd_2g PASSWD_2G]
                                    [--ssid_5g SSID_5G]
                                    [--encryption_5g ENCRYPTION_5G]
                                    [--passwd_5g PASSWD_5G]
                                    [--ssid_6g SSID_6G]
                                    [--encryption_6g ENCRYPTION_6G]
                                    [--passwd_6g PASSWD_6G]
                                    [--log_dur LOG_DUR] [--config_wifi]
                                    [--server_ip SERVER_IP]
                                    [--log_destination LOG_DESTINATION]
                                    [--help_summary]

standard library which supports different functionality of interop
        Operations:

        EXAMPLE-1:
        *    Example of scan results:
        lf_base_interop_profile.py --host 192.168.1.31 --ssid Airtel_9755718444_5GHz --passwd xyz --crypt psk2

        EXAMPLE-2:
        For Wi-Fi Connectivity on all kinds of real devices
        python3 lf_base_interop_profile.py --host 192.168.200.63 --ssid RDT_wpa2 --crypt psk2 --passwd OpenWifi --server_ip 192.16

        NAME: lf_base_interop_profile.py

        STATUS: BETA RELEASE

        SCRIPT_CLASSIFICATION: Connectivity

        SCRIPT_CATEGORIES: Configuration

        VERIFIED_ON:
        Working date    - 29/01/2024
        Build version   - 5.4.7
        kernel version  - 6.2.16+

        License: Free to distribute and modify. LANforge systems must be licensed.
        Copyright 2023 Candela Technologies Inc.


optional arguments:
  -h, --help            show this help message and exit
  --debug               turn on debugging
  --host HOST, --mgr HOST
                        specify the GUI to connect to, assumes port 8080
  --ssid SSID           APPLY: SSID for Interop device WiFi connection
  --crypt CRYPT, --enc CRYPT
                        APPLY: Encryption for Interop device WiFi connection
```

```
  --passwd PASSWD, --pw PASSWD
                        APPLY: Password for Interop device WiFi connection
  --ssid_2g SSID_2G     APPLY: 2G SSID for Interop device WiFi connection
  --encryption_2g ENCRYPTION_2G
                        APPLY: 2G Encryption for Interop device WiFi connection
  --passwd_2g PASSWD_2G
                        APPLY: 2G Password for Interop device WiFi connection
  --ssid_5g SSID_5G     APPLY: 5G SSID for Interop device WiFi connection
  --encryption_5g ENCRYPTION_5G
                        APPLY: 5G Encryption for Interop device WiFi connection
  --passwd_5g PASSWD_5G
                        APPLY: 5G Password for Interop device WiFi connection
  --ssid_6g SSID_6G     APPLY: 6G SSID for Interop device WiFi connection
  --encryption_6g ENCRYPTION_6G
                        APPLY: 6G Encryption for Interop device WiFi connection
  --passwd_6g PASSWD_6G
                        APPLY: 6G Password for Interop device WiFi connection
  --log_dur LOG_DUR, --ld LOG_DUR
                        LOG: Gather ADB logs for a duration of this many minutes
  --config_wifi         Specific this flag to do Wi-Fi connectivity on real devices
  --server_ip SERVER_IP
                        Specific the server IP for the Interop App
  --log_destination LOG_DESTINATION, --log_dest LOG_DESTINATION
                        LOG: the filename destination on the LF device where the log file should be storedGive "stdout" to receive
  --help_summary        Show summary of what this script does
```

py-scripts/lf_chamberview_tools.py

```
usage: ./lf_chamberview_tools.py [-h] [--host MGR] [--mgr_port MGR_PORT]
                                 [--lf_user LF_USER] [--lf_passwd LF_PASSWD]
                                 [--instance INSTANCE] [--log_level LOG_LEVEL]
                                 [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                 [--debug] [--help_summary]

           Chamberview Tool to check if chamberview test is running



optional arguments:
  -h, --help            show this help message and exit
  --host MGR, --mgr MGR, --lf_mgr MGR
                        specify the GUI to connect to
  --mgr_port MGR_PORT   specify the GUI to connect to, default 8080
  --lf_user LF_USER     lanforge user name default: lanforge
  --lf_passwd LF_PASSWD
                        lanforge password default: lanforge
  --instance INSTANCE, --scenario INSTANCE
                        chamber view instance to query
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --debug               Legacy debug flag
  --help_summary        Show summary of what this script does
```

py-scripts/lf_cleanup.py

```
usage: lf_cleanup.py [-h] [--mgr MGR] [--resource RESOURCE] [--cxs]
                     [--l3_endp] [--sta] [--port_mgr] [--br] [--misc]
                     [--layer4] [--sanitize] [--sleep SLEEP] [--debug]
                     [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                     [--log_level LOG_LEVEL] [--help_summary]

NAME: lf_cleanup.py

PURPOSE:
    clean up stations, cross connects and endpoints

EXAMPLE:
    clear all stations:
        ./lf_cleanup.py --mgr localhost --resource 1 --sta

    This example will clean the Port Mgr, Layer-3, L3 Endps, and Layer 4-7 LF GUI tabs:
        ./lf_cleanup.py --mgr localhost --resource 1 --sanitize

    clear all cxs and enps:
        ./lf_cleanup.py --mgr localhost --resource 1 --cxs

    clear all endps:
        ./lf_cleanup.py --mgr localhost --resource 1 --endp

    clear all bridges:
        ./lf_cleanup.py --mgr localhost --resource 1 --br

    clear sta with names phy (not wiphy) and 1.1.eth stations:
        ./lf_cleanup.py --mgr localhost --resource 1 --misc

JSON EXAMPLE:
    clear all stations:
        "args": ["--mgr","192.168.30.12","--resource","1","--sta"]

    This example will clean the Port Mgr, Layer-3, L3 Endps, and Layer 4-7 LF GUI tabs:
        "args": ["--mgr","192.168.30.12","--resource","1","--sanitize"]

    clear all cxs and enps:
        "args": ["--mgr","192.168.30.12","--resource","1","--cxs"]

    clear all endps:
        "args": ["--mgr","192.168.30.12","--resource","1","--endp"]

    clear all bridges:
        "args": ["--mgr","192.168.30.12","--resource","1","--br"]

    clear sta with names phy (not wiphy) and 1.1.eth stations:
        "args": ["--mgr","192.168.30.12","--resource","1","--misc"]

SCRIPT_CLASSIFICATION:  Deletion
```

```
SCRIPT_CATEGORIES:  Functional

NOTES:

    The default port is 8080
    The script will only cleanup what is present in the GUI,
     so it will need to iterate multiple times with script

VERIFIED_ON:
    Tested on 03/17/2023:
        kernel version: 5.19.17+
        gui version: 5.4.6

LICENSE:
         Free to distribute and modify. LANforge systems must be licensed.
         Copyright 2023 Candela Technologies Inc



optional arguments:
  -h, --help            show this help message and exit
  --mgr MGR, --lfmgr MGR
                        --mgr <hostname for where LANforge GUI is running>
  --resource RESOURCE, --res RESOURCE
                        --resource <realm resource> to clear a specific resource, or --resource <all> to cleanup all resources
  --cxs                 --cxs, this will clear all the Layer-3 cxs and endps
  --l3_endp             --endp, this will clear all the Layer-3 endps
  --sta                 --sta, this will clear all the stations generated by a script
  --port_mgr            --port_mgr, this will clear all the created ports on the Port Mgr tab (wifi-sta, mac-vlans, vap, br)
  --br                  --br, this will clear all the bridges
  --misc                --misc, this will clear sta with names phy (not wiphy) and 1.1.eth stations
  --layer4              --layer4, this will clear all the created endpoints on the Layer 4-7 LF GUI tab
  --sanitize            --sanitize, this will clear all the created objects on the Layer-3, L3 Endps, Layer 4-7, and Port Mgr LF G
  --sleep SLEEP         sleep at completion of cleanup in seconds --sleep 2
  --debug               enable debugging
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --help_summary        Show summary of what this script does

          Clean up cxs and endpoints
```

py-scripts/lf_client_visualization.py

```
usage: lf_client_visualization.py [-h] [-m MGR] [-o PORT] [--lf_user LF_USER]
                                  [--lf_password LF_PASSWORD]
                                  [-duration TEST_DURATION]
                                  [-live_chart_dur LIVE_CHART_DURATION]
                                  [-time_btw_updates TIME_BTW_UPDATES]
                                  [-max_report_data MAX_REPORT_DATA]
                                  [-report_compression_interval REPORT_COMPRESSION_INTERVAL]
                                  [-delete] [-pull_report] [-stations]
                                  [-stationsbyRad] [-stationsbyBand]
                                  [-stationsbySsid] [-stationsbyBssid]
                                  [-stationsbyMode] [-stationsbyChannel]
                                  [--verbosity VERBOSITY]
                                  [--graph_groups GRAPH_GROUPS]
                                  [--local_report_dir LOCAL_REPORT_DIR]
                                  [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                  [-c CONFIG_NAME] [--raw_line RAW_LINE]
                                  [--raw_lines_file RAW_LINES_FILE]
                                  [--set SET SET] [--test_rig TEST_RIG]
                                  [--log_level LOG_LEVEL] [--help_summary]

NAME: lf_client_visualization.py

PURPOSE: This client_visualization script is used to Monitor the connection status of all the clients for user specified duration.
         This report shows the connection status of all the clients in the test. This information is very useful
         when running long duration tests with 1000s of WiFi clients connecting across various bands, channels and SSIDs.
         The report shows over time counts of number of clients in scanning, connect and IP address acquired states.
         The report also shows number of clients connected over time per SSID, per Channel, per band and per client type

EXAMPLE:
example 1:
 python3 client_visualization.py --mgr 192.168.200.96 --port 8080 --lf_user lanforge --lf_password lanforge  --test_duration 30s

example 2: To delete existing window.
python3 client_visualization.py --mgr 192.168.200.96 --port 8080 --lf_user lanforge --lf_password lanforge  --test_duration 30s  -

   Note: --pull_report == If specified, this will pull reports from lanforge to your code directory,
                    from where you are running this code

Suggested: To have a scenario already built.

SCRIPT_CLASSIFICATION :  Test
SCRIPT_CATEGORIES:   Monitoring,  Functional, Report Generation

STATUS: BETA RELEASE

VERIFIED_ON:
Working date - 19/02/2024
Build version - 5.4.7
kernel version -  6.7.3+

LICENSE:
    Free to distribute and modify. LANforge systems must be licensed.
    Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False

optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
```

```
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -duration TEST_DURATION, --test_duration TEST_DURATION
                        provide the duration in either hours or minutes or seconds,ex: 2h or 30m or 50s
  -live_chart_dur LIVE_CHART_DURATION, --live_chart_duration LIVE_CHART_DURATION
                        provide the duration in either hours or minutes or seconds,ex: 24h or 10m or 50s
                        Charts in the report will be turned into static pictures with this interval,This helps balance the Memory
  -time_btw_updates TIME_BTW_UPDATES, --time_btw_updates TIME_BTW_UPDATES
                        provide the duration in either hours or minutes or seconds,ex: 1h or 1m or 10s
                        Station stats will be collected at this interval,if test duration is longer than 24 hours then it should b
  -max_report_data MAX_REPORT_DATA, --max_report_data MAX_REPORT_DATA
                        Resulting amount of report samples after compression.
                        A higher sample count gives more graph detail but uses more memory.
                        A lower sample count uses less memory and is more appropriate for long duration tests.
  -report_compression_interval REPORT_COMPRESSION_INTERVAL, --report_compression_interval REPORT_COMPRESSION_INTERVAL
                        Time interval to run report compression periodically.Compression activates when Max Report Data is > 0prov
  -delete, --delete_existing_instance
                        If provided , then the existing instance of Client VIsualization will be deleted (by default: False)
  -pull_report, --pull_report
                        pull reports from lanforge (by default: False)
  -stations, --all_stations
                        To enable All stations(by default: True)
  -stationsbyRad, --stations_by_radios
                        To enable stations_by_radios  (by default: False). i.e stations by radios chart will be included in the re
  -stationsbyBand, --stations_by_Band
                        To enable stations_by_Band  (by default: False)  i.e stations by Bands chart will be included in the repor
  -stationsbySsid, --stations_by_SSID
                        To enable stations_by_SSID  (by default: False)  i.e stations by SSID chart will be included in the report
  -stationsbyBssid, --stations_by_Bssid
                        To enable stations_by_Bssid  (by default: False)  i.e stations by BSSIS chart will be included in the repo
  -stationsbyMode, --stations_by_Mode
                        To enable stations_by_Mode  (by default: False)  i.e stations by Mode chart will be included in the report
  -stationsbyChannel, --stations_by_Channel
                        To enable stations_by_Channel  (by default: False)  i.e stations by Channel chart will be included in the
  --verbosity VERBOSITY
                        Specify verbosity of the report values 1 - 11 default 5, decides the amount of data to be shown in the rep
  --graph_groups GRAPH_GROUPS
                        File to save graph groups to
  --local_report_dir LOCAL_REPORT_DIR
                        --local_report_dir <where to pull reports to>  default is '' , i.e reports will be saved in the current lo
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --help_summary        Show summary of what this script does
```

py-scripts/lf_continuous_throughput_test.py

```
usage: lf_continuous_throughput_test [-h] [-m MGR] [-o PORT]
                                     [--lf_user LF_USER]
                                     [--lf_password LF_PASSWORD]
                                     [-i INSTANCE_NAME] [-c CONFIG_NAME] [-r]
                                     [--load_old_cfg] [--enable ENABLE]
                                     [--disable DISABLE] [--set SET SET]
                                     [--raw_line RAW_LINE]
                                     [--raw_lines_file RAW_LINES_FILE]
                                     [--test_rig TEST_RIG]
                                     [--test_tag TEST_TAG] [--json JSON]
                                     [-u UPSTREAM] [--station STATION]
                                     [--dut DUT]
                                     [--download_speed DOWNLOAD_SPEED]
                                     [--upload_speed UPLOAD_SPEED]
                                     [--duration DURATION]
                                     [--verbosity VERBOSITY]
                                     [--graph_groups GRAPH_GROUPS]
                                     [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                                     [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                     [--log_level LOG_LEVEL] [--help_summary]

NAME: lf_continuous_throughput_test.py

PURPOSE: This script is designed to run continuous throughput tests under various scenarios.

EXAMPLE:
        # Sample cli to test Continuous Throughput Test:

        ./lf_continuous_throughput_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge
        --instance_name continuous-throughput-instance --config_name test_con --upstream 1.1.eth1 --dut LISP_VAP_DUT
        --duration 30s --station 1.1.wlan0 --download_speed 85% --upload_speed 0 --raw_line 'pkts: 60'
        --raw_line 'cust_pkt_sz: 88 1200' --raw_line 'directions: DUT Transmit' --raw_line 'traffic_types: UDP'
        --raw_line 'bandw_options: 20' --raw_line 'spatial_streams: 2' --raw_line 'modes: 802.11bgn-AX' --pull_report

        # Sample cli to test continuous_throughput Test with influx db (Optional):

        ./lf_continuous_throughput_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge
        --instance_name continuous-throughput-instance --config_name test_con --upstream 1.1.eth2 --dut linksys-8450
        --duration 15s --station 1.1.sta01500 --download_speed 85% --upload_speed 0
        --raw_line 'pkts: Custom;60;142;256;512;1024;MTU' --raw_line 'cust_pkt_sz: 88 1200'
        --raw_line 'directions: DUT Transmit;DUT Receive' --raw_line 'traffic_types: UDP;TCP'
        --test_rig Testbed-01 --pull_report

Example 2:

        # Sample cli to test Continuous Throughput Test with <_ct_cli_config_>.json:

        ./lf_continuous_throughput_test.py --json <name>.json

        The Example/Sample json file should be:
```

```
            "lf_continuous_throughput_config.json"

            Sample <name>.json between using eth1 and eth2
            {
                "mgr":"192.168.0.101",
                "port":"8080",
                "lf_user":"lanforge",
                "lf_password":"lanforge",
                "instance_name":"continuous-throughput-instance",
                "config_name":"test_con",
                "upstream":"1.1.eth1",
                "dut":"asus_5g",
                "duration":"15s",
                "station":"1.1.eth2",
                "download_speed":"85%",
                "upload_speed":"0",
                "raw_line":  ["pkts: Custom;60;MTU", "cust_pkt_sz: 88 1200", "directions: DUT Transmit",
                "traffic_types: UDP", "bandw_options: 20", "spatial_streams: 1"]
            }

            Sample <name>.json between using eth1 and station 1.1.sta0002
            {
                "mgr":"192.168.0.101",
                "port":"8080",
                "lf_user":"lanforge",
                "lf_password":"lanforge",
                "instance_name":"continuous-throughput-instance",
                "config_name":"test_con",
                "upstream":"1.1.eth1",
                "dut":"asus_5g",
                "duration":"15s",
                "station":"1.1.sta0002",
                "download_speed":"85%",
                "upload_speed":"0",
                "raw_line":  ["pkts: Custom;60;MTU", "cust_pkt_sz: 88 1200", "directions: DUT Transmit",
                "traffic_types: UDP", "bandw_options: 20", "spatial_streams: 1"]
            }

SCRIPT_CLASSIFICATION:  Test

SCRIPT_CATEGORIES:   Performance,  Functional,  KPI Generation,  Report Generation

NOTES:
        This script is used to automate running continuous_throughput tests.  You may need to view
        a continuous_throughput test configured through the GUI to understand the options and how best to input data.

        Note :
                To Run this script gui should be opened with

                path: cd LANforgeGUI_5.4.7 (5.4.7 can be changed with GUI version)
                        pwd (Output : /home/lanforge/LANforgeGUI_5.4.7)
                        ./lfclient.bash -cli-socket 3990

        ---> lf_continuous_throughput_test.py is designed to run continuous_throughput tests under various scenarios.

            ./lf_continuous_throughput_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge
            --instance_name <instance name> --config_name test_con --upstream <upstream port> --dut <dut name>
            --duration <test duration> --station <staion name> --download_speed <download rate>
            --upload_speed <opposite rate>
            --raw_line 'pkts: 60' --raw_line 'cust_pkt_sz: 88 1200' --raw_line 'directions: DUT Transmit'
            --raw_line 'traffic_types: UDP' --raw_line 'bandw_options: 20' --raw_line 'spatial_streams: 2'
            --raw_line 'modes: 802.11bgn-AX' --pull_report

            *   --raw_line : 'line contents' will add any setting to the test config. This is useful way to support
                        any options not specifically enabled by the command options.

            *  --set modifications will be applied after the other config has happened, so it can be used to
                        override any other config.

    Example of raw text config for continuous_throughput, to show other possible options:

    show_events: 1
    show_log: 0
    log_stdout: 0
    get_csv_cb: 0
    auto_save: 1
    rpt_path:
    rpt_path_make_subdir: 1
    test_rig:
    test_tag:
    rpt_name:
    rpt_dir_prefix_textfield:
    show_scan: 1
    auto_helper: 0
    allow_llw: 0
    use_mcs_histograms: 1
    disable_mlo: 0
    extra_tx_status: 0
    extra_rx_status: 0
    txs_all_status: 0
    sae_pwe: 2
    skip_ac: 0
    skip_be: 0
    skip_ax: 0
    skip_2: 0
    skip_6: 0
    skip_5: 0
    skip_5b: 1
    skip_dual: 0
    skip_tri: 1
    skip_rebuild: 0
    udp_gro: 0
    multi_cx: 1
    udp_burst: 0
    payload_pattern: 0
    sndbuf_tcp: 0
    sndbuf_udp: 0
```

```
    rcvbuf_tcp: 0
    rcvbuf_udp: 0
    show_per_loop_totals: 1
    clear_ports_too: 0
    show_log_scale: 0
    realtime_per_port: 0
    realtime_per_cx: 0
    selected_dut: eero-root
    rvr_bringup_wait: 30000
    first_byte_wait: 30000
    duration: 300000
    settle_time: 10000
    sndbuf: 0
    rvr_helper:
    rcvbuf: 0
    traffic_port: 1.1.wlan0
    upstream_port: 1.1.eth3
    path_loss: 10
    ap_txpower: 0
    speed: 85%
    speed2: 0
    min_rssi_bound: -150
    max_rssi_bound: 0
    channels: AUTO
    modes: Auto
    pkts: Custom;60;MTU
    spatial_streams: 1
    security_options: AUTO
    bandw_options: AUTO
    traffic_types: UDP
    directions: DUT Transmit
    txo_preamble: OFDM
    txo_mcs: 0 CCK, OFDM, HT, VHT
    txo_retries: No Retry
    txo_sgi: OFF
    txo_txpower: 17
    attenuator: 0
    attenuator2: 0
    attenuator_mod: 255
    attenuator_mod2: 255
    attenuations: 0..+50..950
    attenuations2: 0..+50..950
    chamber: Root
    cust_pkt_sz: 88 1200
    show_bar_labels: 1
    show_prcnt_tput: 0
    show_passfail: 1
    show_last: 0
    show_gp_graphs: 1
    show_polar_graphs: 1
    show_line_graphs: 1
    pause_iter: 0
    max_unused_atten: 0
    adb_modify_wifi: 0
    aggregate_arc_count: 4
    show_realtime: 1
    mconn: 1
    mpkt: 1000
    tos: 0
    loop_iterations: 1

STATUS: Functional

VERIFIED_ON: 05-June-2024,
             GUI Version:  5.4.8
             Kernel Version: 6.9.0+

LICENSE:
          Free to distribute and modify. LANforge systems must be licensed.
          Copyright 2024 Candela Technologies Inc

INCLUDE_IN_README: False



optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the capacity test?  Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).  See example raw text config for possible options.  M
  --disable DISABLE     Specify options to disable (set value to 0).  See example raw text config for possible options.  May be sp
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for instance:  testbed-01
  --json JSON           Path to JSON configuration file for test. When specified, JSON takes precedence over command line args.
  -u UPSTREAM, --upstream UPSTREAM
                        Upstream port used in test. For example, '1.1.eth2'
  --station STATION     Station used in test. Example: '1.1.sta01500'
  --dut DUT             Name of DUT used in test. Assumes DUT is already configured in LANforge. Example: 'linksys-8450'
  --download_speed DOWNLOAD_SPEED
                        Requested download speed used in test. Percentage of theoretical is also supported. Default: 85%.
  --upload_speed UPLOAD_SPEED
                        Requested upload speed used in test. Percentage of theoretical is also supported. Default: 0
  --duration DURATION   Duration of each traffic run
  --verbosity VERBOSITY
```

```
                        Verbosity of the report specified as single value in 1 - 11 range (whole numbers).
                        The larger the number, the more verbose. Default: 5
  --graph_groups GRAPH_GROUPS
                        Path to file to save graph_groups to on local system
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        Path to directory to pull remote report data to on local system
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        Path to logger JSON configuration
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --help_summary        Show summary of what this script does

        Continuous Throughput Test
```

py-scripts/lf_create_vap_cv.py

```
usage: lf_create_vap_cv.py [-h] [-m MGR] [-o PORT] [--lf_user LF_USER]
                           [--lf_password LF_PASSWORD] [-i INSTANCE_NAME]
                           [-c CONFIG_NAME] [-r] [--load_old_cfg]
                           [--enable ENABLE] [--disable DISABLE]
                           [--set SET SET] [--raw_line RAW_LINE]
                           [--raw_lines_file RAW_LINES_FILE]
                           [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                           [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                           [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                           [--log_level LOG_LEVEL] [-dos] [-sn SCENARIO_NAME]
                           [-vr VAP_RADIO] [-vf VAP_FREQ] [-vs VAP_SSID]
                           [-vp VAP_PASSWD] [-vse VAP_SECURITY]
                           [--vap_upstream_port VAP_UPSTREAM_PORT]
                           [--num_vaps NUM_VAPS] [--vap_bw VAP_BW]
                           [--vap_mode VAP_MODE] [--set_upstream SET_UPSTREAM]
                           [--vap_ip VAP_IP] [--vap_ip_mask VAP_IP_MASK]
                           [--gateway GATEWAY]
                           [--dhcp_min_range DHCP_MIN_RANGE]
                           [--dhcp_max_range DHCP_MAX_RANGE]
                           [--load_existing_scenario]
                           [--old_scenario_name OLD_SCENARIO_NAME]
                           [--help_summary]

NAME: lf_create_vap_cv.py

PURPOSE:
    This script will create a vap using chamberview based upon a user defined frequency.

EXAMPLE:
    Use './lf_create_vap_cv.py --help' to see command line usage and options

    ./lf_create_vap_cv.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge
        --delete_old_scenario --scenario_name Automation --vap_radio wiphy0 --set_upstream True
        --vap_freq 2437 --vap_ssid routed-AP --vap_passwd something --vap_security wpa2 --vap_bw 20

    # For modify the net-smith dhcp min and max range
    python3 ./lf_create_vap_cv.py --mgr 192.168.200.138 --port 8080 --delete_old_scenario --scenario_name hello
    --vap_radio 1.1.wiphy0 --vap_freq 2437 --vap_ssid testings --vap_passwd Password@123 --vap_security wpa2 --num_vaps 2
    --vap_bw 20 --vap_ip 192.168.0.16 --vap_ip_mask 255.255.0.0 --dhcp_min_range 192.168.0.5 --dhcp_max_range 192.168.0.200

    vs_code launch.json example:
    "args": ["--mgr","localhost",
             "--port","8080",
             "--lf_user","lanforge",
             "--lf_password","lanforge",
             "--vap_radio","wiphy0",
             "--vap_freq","36",
             "--vap_ssid","test_vap",
             "--vap_passwd","password",
             "--vap_security","wpa2",
             "--vap_upstream_port","1.1.eth1"
            ]

            "args": ["--mgr","192.168.0.104",
            "--port","8080",
            "--lf_user","lanforge",
            "--lf_password","lanforge",
            "--delete_old_scenario",
            "--scenario_name","dfs",
            "--vap_radio","wiphy3",
            "--vap_freq","5660",
            "--vap_ssid","mtk7915_5g",
            "--vap_passwd","lf_mtk7915_5g",
            "--vap_security","wpa2",
            "--vap_bw","20",
            "--vap_upstream_port","1.1.eth2"
            ]

SCRIPT_CLASSIFICATION:  Creation

SCRIPT_CATEGORIES: Functional

NOTES:

This script creates
1. Chamber view scenario for vap
2. Vap profile with given parameters

STATUS:   BETA RELEASE

VERIFIED_ON:
Working date : 05-July-2023
Build version: 5.4.6
Kernel version: 6.2.16+

LICENSE:
    Free to distribute and modify. LANforge systems must be licensed.
    Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False
```

```
optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the capacity test?  Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).  See example raw text config for possible options.  M
  --disable DISABLE     Specify options to disable (set value to 0).  See example raw text config for possible options.  May be sp
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for instance:  testbed-01
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir <where to pull reports to>  default '' put where dataplane script run from
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  -dos, --delete_old_scenario
                        To delete old scenarios (by default: True)
  -sn SCENARIO_NAME, --scenario_name SCENARIO_NAME
                        Chamberview scenario name (by default: Automation
  -vr VAP_RADIO, --vap_radio VAP_RADIO
                        vap radio name (by default: wiphy0
  -vf VAP_FREQ, --vap_freq VAP_FREQ
                        vap frequency (by default: 2437
  -vs VAP_SSID, --vap_ssid VAP_SSID
                        vap ssid (by default: routed-AP
  -vp VAP_PASSWD, --vap_passwd VAP_PASSWD
                        vap password (by default:None
  -vse VAP_SECURITY, --vap_security VAP_SECURITY
                        vap security like wep ,wpa, wpa2, wpa3 (by default: wpa2
  --vap_upstream_port VAP_UPSTREAM_PORT
                        vap upstream_port (by default: 1.1.eth2
  --num_vaps NUM_VAPS   enter the number of vaps need to be created (by default : 1)
  --vap_bw VAP_BW       vap bw like 20, 40, 80, 160(by default: None
  --vap_mode VAP_MODE   vap mode can be selected from these"AUTO", "a", "aAX", "abg", "abgn", "abgnAC", "abgnAX", "an","anAC", "an
  --set_upstream SET_UPSTREAM
                        Enter True if upstream need to be set else enter False
  --vap_ip VAP_IP       Modify the VAP DHCP ip address
  --vap_ip_mask VAP_IP_MASK
                        Modify the VAP ip mask
  --gateway GATEWAY     Modify the VAP Gateway ip
  --dhcp_min_range DHCP_MIN_RANGE
                        Modify the VAP DHCP min range
  --dhcp_max_range DHCP_MAX_RANGE
                        Modify the VAP DHCP max range
  --load_existing_scenario
                        select if you want to load existing scenario
  --old_scenario_name OLD_SCENARIO_NAME
                        provide old scenario name to be loaded
  --help_summary        Show summary of what this script does
```

py-scripts/lf_create_wanlink.py

```
usage: ./lf_create_wanlink.py [-h] [--host MGR] [--mgr_port MGR_PORT]
                              [--lf_user LF_USER] [--lf_passwd LF_PASSWD]
                              [--wl_name WL_NAME] [--cpu_id CPU_ID]
                              [--description DESCRIPTION] [--latency LATENCY]
                              [--latency_A LATENCY_A] [--latency_B LATENCY_B]
                              [--max_rate MAX_RATE] [--max_rate_A MAX_RATE_A]
                              [--max_rate_B MAX_RATE_B] [--port_A PORT_A]
                              [--port_B PORT_B] [--resource RESOURCE]
                              [--shelf SHELF] [--wle_flags WLE_FLAGS]
                              [--drop_freq DROP_FREQ]
                              [--drop_freq_A DROP_FREQ_A]
                              [--drop_freq_B DROP_FREQ_B]
                              [--dup_freq DUP_FREQ] [--dup_freq_A DUP_FREQ_A]
                              [--dup_freq_B DUP_FREQ_B]
                              [--extra_buffer EXTRA_BUFFER]
                              [--extra_buffer_A EXTRA_BUFFER_A]
                              [--extra_buffer_B EXTRA_BUFFER_B]
                              [--jitter_freq JITTER_FREQ]
                              [--jitter_freq_A JITTER_FREQ_A]
                              [--jitter_freq_B JITTER_FREQ_B]
                              [--latency_packet LATENCY_PACKET]
                              [--latency_packet_A LATENCY_PACKET_A]
                              [--latency_packet_B LATENCY_PACKET_B]
                              [--max_drop_amt MAX_DROP_AMT]
                              [--max_drop_amt_A MAX_DROP_AMT_A]
                              [--max_drop_amt_B MAX_DROP_AMT_B]
                              [--max_jitter MAX_JITTER]
                              [--max_jitter_A MAX_JITTER_A]
                              [--max_jitter_B MAX_JITTER_B]
                              [--max_lateness MAX_LATENESS]
                              [--max_lateness_A MAX_LATENESS_A]
                              [--max_lateness_B MAX_LATENESS_B]
                              [--max_reorder_amt MAX_REORDER_AMT]
                              [--max_reorder_amt_A MAX_REORDER_AMT_A]
                              [--max_reorder_amt_B MAX_REORDER_AMT_B]
                              [--min_drop_amt MIN_DROP_AMT]
                              [--min_drop_amt_A MIN_DROP_AMT_A]
                              [--min_drop_amt_B MIN_DROP_AMT_B]
                              [--min_reorder_amt MIN_REORDER_AMT]
                              [--min_reorder_amt_A MIN_REORDER_AMT_A]
```

```
                            [--min_reorder_amt_B MIN_REORDER_AMT_B]
                            [--playback_capture_file PLAYBACK_CAPTURE_FILE]
                            [--reorder_freq REORDER_FREQ]
                            [--reorder_freq_A REORDER_FREQ_A]
                            [--reorder_freq_B REORDER_FREQ_B]
                            [--speed SPEED] [--speed_A SPEED_A]
                            [--speed_B SPEED_B]
                            [--drop_nth_pkt DROP_NTH_PKT]
                            [--drop_nth_pkt_A DROP_NTH_PKT_A]
                            [--drop_nth_pkt_B DROP_NTH_PKT_B]
                            [--suppress_related_commands] [--kernel_mode]
                            [--pass_through_mode] [--log_level LOG_LEVEL]
                            [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                            [--debug] [--help_summary]

NAME: lf_create_wanlink.py

PURPOSE: creates a wanlink using the lanforge api.

EXAMPLE:
Both port_A and port_B have the same configuraiton
$ ./lf_create_wanlink.py --mgr 192.168.0.104 --mgr_port 8080 --port_A eth1 --port_B eth2    --speed 1024000 --wl_name wanlink --la

Mixed configuration for port_A and port_B
$ ./lf_create_wanlink.py --mgr 192.168.0.104 --mgr_port 8080 --port_A eth1 --port_B eth2    --speed_A 1024000 --speed_B 2048000 --

NOTES:
ip-address must be assigned to the wanlink endpoints in the LANforge gui for scenario to run.



optional arguments:
  -h, --help            show this help message and exit
  --host MGR, --mgr MGR
                        specify the GUI to connect to
  --mgr_port MGR_PORT   specify the GUI to connect to, default 8080
  --lf_user LF_USER     lanforge user name default lanforge
  --lf_passwd LF_PASSWD
                        lanforge password defualt lanforge
  --wl_name WL_NAME, --alias WL_NAME
                        (add wl endp) The name of the endpoint we are configuring. [R]
  --cpu_id CPU_ID       (add wl endp) The CPU/thread that this process should run on (kernel-mode only). Default = 'NA'
  --description DESCRIPTION
                        (add wl endp) Description for this endpoint, put in single quotes if it contains spaces Default = 'NA'
  --latency LATENCY     (add wl endp) The latency (ms) that will be added to each packet entering this WanLink. Default = 'NA' bot
  --latency_A LATENCY_A
                        (add wl endp) The latency (ms) that will be added to each packet entering this WanLink. Default = None por
  --latency_B LATENCY_B
                        (add wl endp) The latency (ms) that will be added to each packet entering this WanLink. Default = None por
  --max_rate MAX_RATE   (add wl endp) Use --speed parameter instead. Maximum transmit rate (bps) for this WanLink. Default = 10240
  --max_rate_A MAX_RATE_A
                        (add wl endp) Use --speed_A parameter instead. Maximum transmit rate (bps) for this WanLink. Default = Non
  --max_rate_B MAX_RATE_B
                        (add wl endp) Use --speed_B parameter instead. Maximum transmit rate (bps) for this WanLink. Default = Non
  --port_A PORT_A       (add wl endp) Endpoint A
  --port_B PORT_B       (add wl endp) Endpoint B
  --resource RESOURCE   (add wl endp) LANforge resource Default
  --shelf SHELF         (add wl endp) LANforge Shelf name/id
  --wle_flags WLE_FLAGS
                        (add wl endp) WanLink Endpoint specific flags, Default = 1, SHOW_WP = 1 .Show WanPaths in wanlink endpoint
  --drop_freq DROP_FREQ
                        (set wanlink info) How often, out of 1,000,000 packets, should we purposefully drop a packet. Default = 0
  --drop_freq_A DROP_FREQ_A
                        (set wanlink info) How often, out of 1,000,000 packets, should we purposefully drop a packet. Default = No
  --drop_freq_B DROP_FREQ_B
                        (set wanlink info) How often, out of 1,000,000 packets, should we purposefully drop a packet. Default = No
  --dup_freq DUP_FREQ   (set wanlink info) How often, out of 1,000,000 packets, should we purposefully duplicate a packet. Default
  --dup_freq_A DUP_FREQ_A
                        (set wanlink info) How often, out of 1,000,000 packets, should we purposefully duplicate a packet. Default
  --dup_freq_B DUP_FREQ_B
                        (set wanlink info) How often, out of 1,000,000 packets, should we purposefully duplicate a packet. Default
  --extra_buffer EXTRA_BUFFER
                        (set wanlink info) The extra amount of bytes to buffer before dropping pkts, in units of 1024. Use -1 for
  --extra_buffer_A EXTRA_BUFFER_A
                        (set wanlink info) The extra amount of bytes to buffer before dropping pkts, in units of 1024. Use -1 for
  --extra_buffer_B EXTRA_BUFFER_B
                        (set wanlink info) The extra amount of bytes to buffer before dropping pkts, in units of 1024. Use -1 for
  --jitter_freq JITTER_FREQ
                        (set wanlink info) How often, out of 1,000,000 packets, should we apply jitter. Default = 0 both ports (%)
  --jitter_freq_A JITTER_FREQ_A
                        (set wanlink info) How often, out of 1,000,000 packets, should we apply jitter. Default = None port A (%)
  --jitter_freq_B JITTER_FREQ_B
                        (set wanlink info) How often, out of 1,000,000 packets, should we apply jitter. Default = None port B (%)
  --latency_packet LATENCY_PACKET
                        (set wanlink info) The base latency added to all packets, in milliseconds (or add 'us' suffix for microsec
  --latency_packet_A LATENCY_PACKET_A
                        (set wanlink info) The base latency added to all packets, in milliseconds (or add 'us' suffix for microsec
  --latency_packet_B LATENCY_PACKET_B
                        (set wanlink info) The base latency added to all packets, in milliseconds (or add 'us' suffix for microsec
  --max_drop_amt MAX_DROP_AMT
                        (set wanlink info) Maximum amount of packets to drop in a row. Default is 1. both ports
  --max_drop_amt_A MAX_DROP_AMT_A
                        (set wanlink info) Maximum amount of packets to drop in a row. Default is None. port A
  --max_drop_amt_B MAX_DROP_AMT_B
                        (set wanlink info) Maximum amount of packets to drop in a row. Default is None. port B
  --max_jitter MAX_JITTER
                        (set wanlink info) The maximum jitter, in milliseconds (or ad 'us' suffix for microseconds) Default = 10 b
  --max_jitter_A MAX_JITTER_A
                        (set wanlink info) The maximum jitter, in milliseconds (or ad 'us' suffix for microseconds) port A (ms)
  --max_jitter_B MAX_JITTER_B
                        (set wanlink info) The maximum jitter, in milliseconds (or ad 'us' suffix for microseconds) port B (ms)
  --max_lateness MAX_LATENESS
                        (set wanlink info) Maximum amount of un-intentional delay before pkt both ports (ms) is dropped. Default i
  --max_lateness_A MAX_LATENESS_A
                        (set wanlink info) Maximum amount of un-intentional delay before pkt both ports (ms) is dropped. Default i
  --max_lateness_B MAX_LATENESS_B
                        (set wanlink info) Maximum amount of un-intentional delay before pkt both ports (ms) is dropped. Default i
```

```
  --max_reorder_amt MAX_REORDER_AMT
                        (set wanlink info) Maximum amount of packets by which to reorder, Default is 10. both ports (ms)
  --max_reorder_amt_A MAX_REORDER_AMT_A
                        (set wanlink info) Maximum amount of packets by which to reorder, Default is 10. both ports (ms) port A (m
  --max_reorder_amt_B MAX_REORDER_AMT_B
                        (set wanlink info) Maximum amount of packets by which to reorder, Default is 10. both ports (ms) port B (m
  --min_drop_amt MIN_DROP_AMT
                        (set wanlink info) Minimum amount of packets to drop in a row. Default is 1. both ports (ms)
  --min_drop_amt_A MIN_DROP_AMT_A
                        (set wanlink info) Minimum amount of packets to drop in a row. Default is 1. both ports (ms) port A (ms)
  --min_drop_amt_B MIN_DROP_AMT_B
                        (set wanlink info) Minimum amount of packets to drop in a row. Default is 1. both ports (ms) port B (ms)
  --min_reorder_amt MIN_REORDER_AMT
                        (set wanlink info) Minimum amount of packets by which to reorder, Default is 1. both ports
  --min_reorder_amt_A MIN_REORDER_AMT_A
                        (set wanlink info) Minimum amount of packets by which to reorder, Default is 1. port A
  --min_reorder_amt_B MIN_REORDER_AMT_B
                        (set wanlink info) Minimum amount of packets by which to reorder, Default is 1. port B
  --playback_capture_file PLAYBACK_CAPTURE_FILE
                        (set wanlink info) Name of the WAN capture file to play back. Default = None
  --reorder_freq REORDER_FREQ
                        (set wanlink info) How often, out of 1,000,000 packets, should we make a packet out of order. both ports D
  --reorder_freq_A REORDER_FREQ_A
                        (set wanlink info) How often, out of 1,000,000 packets, should we make a packet out of order. port A Defau
  --reorder_freq_B REORDER_FREQ_B
                        (set wanlink info) How often, out of 1,000,000 packets, should we make a packet out of order. port B Defau
  --speed SPEED         (set wanlink info) Use this instead of max_rate. The maximum speed of traffic this endpoint will accept (b
  --speed_A SPEED_A     (set wanlink info) Use this instead of max_rate_A. The maximum speed of traffic this endpoint will accept
  --speed_B SPEED_B     (set wanlink info) Use this instead of max_rate_B. The maximum speed of traffic this endpoint will accept
  --drop_nth_pkt DROP_NTH_PKT
                        (set wanlink info) drop packets at every Nth received packet
  --drop_nth_pkt_A DROP_NTH_PKT_A
                        (set wanlink info) drop packets at every Nth received packet, port A
  --drop_nth_pkt_B DROP_NTH_PKT_B
                        (set wanlink info) drop packets at every Nth received packet, port B
  --suppress_related_commands
                        (set wanlink info) Used by lanforge_api Default False if set store true
  --kernel_mode         (set endp flag) Select  kernel-mode Wanlinks, must be the same for both endpoint sets both ports, Default
  --pass_through_mode
                                (set endp flag) pass-through means disable all impairments and slow-downs, without having to manua
                                Default = False', action='store_true'

  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --debug               Legacy debug flag
  --help_summary        Show summary of what this script does
```

py-scripts/lf_csv.py

```
usage: lf_mixed_traffic.py [-h] [--help_summary]


NAME: lf_csv.py

PURPOSE:
        The purpose of this test is to create a csv file for the lanforge output of a test
        This file should should be imported by the file running the test and data can be passed while creating the object

EXAMPLE:
        # CLI To run the lf_csv to generate the csv file
        python3 lf_csv.py


optional arguments:
  -h, --help      show this help message and exit
  --help_summary  Show summary of what this script does

               The lf_csv is used to generate csv report of the lanforge test results
```

py-scripts/lf_dataplane_test.py

Example report: dataplane.pdf

```
usage: lf_dataplane_test [-h] [-m MGR] [-o PORT] [--lf_user LF_USER]
                         [--lf_password LF_PASSWORD] [-i INSTANCE_NAME]
                         [-c CONFIG_NAME] [-r] [--load_old_cfg]
                         [--enable ENABLE] [--disable DISABLE] [--set SET SET]
                         [--raw_line RAW_LINE]
                         [--raw_lines_file RAW_LINES_FILE]
                         [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                         [--json JSON] [-u UPSTREAM] [--station STATION]
                         [--dut DUT] [--download_speed DOWNLOAD_SPEED]
                         [--upload_speed UPLOAD_SPEED] [--duration DURATION]
                         [--verbosity VERBOSITY] [--graph_groups GRAPH_GROUPS]
                         [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                         [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                         [--help_summary]

NAME: lf_dataplane_test.py

PURPOSE: This script is designed to run dataplane tests under various scenarios.

EXAMPLE:
        # Sample cli to test Dataplane Test :

        ./lf_dataplane_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge
        --instance_name dataplane-instance --config_name test_con --upstream 1.1.eth1 --dut LISP_VAP_DUT
        --duration 30s --station 1.1.wlan0 --download_speed 85% --upload_speed 0 --raw_line 'pkts: 60'
        --raw_line 'cust_pkt_sz: 88 1200' --raw_line 'directions: DUT Transmit' --raw_line 'traffic_types: UDP'
        --raw_line 'bandw_options: 20' --raw_line 'spatial_streams: 2' --raw_line 'modes: 802.11bgn-AX' --pull_report

Example 2:

        # Sample cli to test Dataplane Test with <_dp_cli_config_>.json :
```

```
        ./lf_dataplane_test.py --json <name>.json

        The Example/Sample json file should be :

            "lf_dataplane_config.json"

            Sample <name>.json between using eth1 and eth2
            {
                "mgr":"192.168.0.101",
                "port":"8080",
                "lf_user":"lanforge",
                "lf_password":"lanforge",
                "instance_name":"dataplane-instance",
                "config_name":"test_con",
                "upstream":"1.1.eth1",
                "dut":"asus_5g",
                "duration":"15s",
                "station":"1.1.eth2",
                "download_speed":"85%",
                "upload_speed":"0",
                "raw_line":  ["pkts: Custom;60;MTU", "cust_pkt_sz: 88 1200", "directions: DUT Transmit",
                "traffic_types: UDP", "bandw_options: 20", "spatial_streams: 1"]
            }

            Sample <name>.json between using eth1 and station 1.1.sta0002
            {
                "mgr":"192.168.0.101",
                "port":"8080",
                "lf_user":"lanforge",
                "lf_password":"lanforge",
                "instance_name":"dataplane-instance",
                "config_name":"test_con",
                "upstream":"1.1.eth1",
                "dut":"asus_5g",
                "duration":"15s",
                "station":"1.1.sta0002",
                "download_speed":"85%",
                "upload_speed":"0",
                "raw_line":  ["pkts: Custom;60;MTU", "cust_pkt_sz: 88 1200", "directions: DUT Transmit",
                "traffic_types: UDP", "bandw_options: 20", "spatial_streams: 1"]
            }

SCRIPT_CLASSIFICATION:  Test

SCRIPT_CATEGORIES:   Performance,  Functional,  KPI Generation,  Report Generation

NOTES:
        This script is used to automate running Dataplane tests.  You may need to view a Dataplane test
        configured through the GUI to understand the options and how best to input data.

        Note :
                To Run this script gui should be opened with

                path: cd LANforgeGUI_5.4.3 (5.4.3 can be changed with GUI version)
                        pwd (Output : /home/lanforge/LANforgeGUI_5.4.3)
                        ./lfclient.bash -cli-socket 3990

        ---> lf_dataplane_test.py is designed to run dataplane tests under various scenarios.

            ./lf_dataplane_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge
            --instance_name <instance name> --config_name test_con --upstream <upstream port> --dut <dut name>
            --duration <test duration> --station <staion name> --download_speed <download rate> --upload_speed <Opposit rate>
            --raw_line 'pkts: 60' --raw_line 'cust_pkt_sz: 88 1200' --raw_line 'directions: DUT Transmit'
            --raw_line 'traffic_types: UDP' --raw_line 'bandw_options: 20' --raw_line 'spatial_streams: 2'
            --raw_line 'modes: 802.11bgn-AX' --pull_report

            *   --raw_line : 'line contents' will add any setting to the test config. This is useful way to support
                        any options not specifically enabled by the command options.

            *  --set modifications will be applied after the other config has happened, so it can be used to
                        override any other config.

    Example of raw text config for Dataplane, to show other possible options:

    show_events: 1
    show_log: 0
    port_sorting: 0
    kpi_id: Dataplane Pkt-Size
    notes0: ec5211 in bridge mode, wpa2 auth.
    bg: 0xE0ECF8
    test_rig:
    show_scan: 1
    auto_helper: 0
    skip_2: 0
    skip_5: 0
    skip_5b: 1
    skip_dual: 0
    skip_tri: 1
    selected_dut: ea8300
    duration: 15000
    traffic_port: 1.1.157 sta01500
    upstream_port: 1.1.2 eth2
    path_loss: 10
    speed: 85%
    speed2: 0Kbps
    min_rssi_bound: -150
    max_rssi_bound: 0
    channels: AUTO
    modes: Auto
    pkts: Custom;60;142;256;512;1024;MTU
    spatial_streams: AUTO
    security_options: AUTO
    bandw_options: AUTO
    traffic_types: UDP;TCP
    directions: DUT Transmit;DUT Receive
    txo_preamble: OFDM
    txo_mcs: 0 CCK, OFDM, HT, VHT
```

```
    txo_retries: No Retry
    txo_sgi: OFF
    txo_txpower: 15
    attenuator: 0
    attenuator2: 0
    attenuator_mod: 255
    attenuator_mod2: 255
    attenuations: 0..+50..950
    attenuations2: 0..+50..950
    chamber: 0
    tt_deg: 0..+45..359
    cust_pkt_sz: 88 1200
    show_bar_labels: 1
    show_prcnt_tput: 0
    show_3s: 0
    show_ll_graphs: 0
    show_gp_graphs: 1
    show_lm: 1
    pause_iter: 0
    outer_loop_atten: 0
    show_realtime: 1
    operator:
    mconn: 1
    mpkt: 1000
    tos: 0
    loop_iterations: 1

STATUS: Functional

VERIFIED_ON:   11-MAY-2023,
               GUI Version:  5.4.6
               Kernel Version: 6.2.14+

LICENSE:
          Free to distribute and modify. LANforge systems must be licensed.
          Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False



optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the capacity test?  Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).  See example raw text config for possible options.  M
  --disable DISABLE     Specify options to disable (set value to 0).  See example raw text config for possible options.  May be sp
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for instance:  testbed-01
  --json JSON           Path to JSON configuration file for test. When specified, JSON takes precedence over command line args.
  -u UPSTREAM, --upstream UPSTREAM
                        Upstream port used in test. For example, '1.1.eth2'
  --station STATION     Station used in test. Example: '1.1.sta01500'
  --dut DUT             Name of DUT used in test. Assumes DUT is already configured in LANforge. Example: 'linksys-8450'
  --download_speed DOWNLOAD_SPEED
                        Requested download speed used in test. Percentage of theoretical is also supported. Default: 85%.
  --upload_speed UPLOAD_SPEED
                        Requested upload speed used in test. Percentage of theoretical is also supported. Default: 0
  --duration DURATION   Duration of each traffic run
  --verbosity VERBOSITY
                        Verbosity of the report specified as single value in 1 - 11 range (whole numbers).
                        The larger the number, the more verbose. Default: 5
  --graph_groups GRAPH_GROUPS
                        Path to file to save graph_groups to on local system
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        Path to directory to pull remote report data to on local system
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        Path to logger JSON configuration
  --help_summary        Show summary of what this script does

         Data Plane Test
```

py-scripts/lf_ftp.py

```
usage: lf_ftp.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                 [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                 [--upstream_port UPSTREAM_PORT] [--ssid SSID]
                 [--passwd PASSWD] [--security SECURITY] [--ap_name AP_NAME]
                 [--ap_ip AP_IP] [--twog_radio TWOG_RADIO]
                 [--fiveg_radio FIVEG_RADIO] [--sixg_radio SIXG_RADIO]
                 [--lf_username LF_USERNAME] [--lf_password LF_PASSWORD]
                 [--traffic_duration TRAFFIC_DURATION]
                 [--clients_type CLIENTS_TYPE] [--dowebgui DOWEBGUI]
                 [--ssh_port SSH_PORT] [--bands BANDS [BANDS ...]]
                 [--directions DIRECTIONS [DIRECTIONS ...]]
                 [--file_sizes FILE_SIZES [FILE_SIZES ...]]
                 [--num_stations NUM_STATIONS] [--result_dir RESULT_DIR]
                 [--device_list DEVICE_LIST] [--test_name TEST_NAME]
                 [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                 [--dut_hw_version DUT_HW_VERSION]
                 [--dut_sw_version DUT_SW_VERSION]
                 [--dut_model_num DUT_MODEL_NUM]
                 [--dut_serial_num DUT_SERIAL_NUM]
```

```
                [--test_priority TEST_PRIORITY] [--test_id TEST_ID]
                [--csv_outfile CSV_OUTFILE]
                [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                [--help_summary]

---------------------------

NAME: lf_ftp.py

PURPOSE:
lf_ftp.py will verify that N clients are connected on a specified band and can simultaneously download/upload
some amount of file data from the FTP server while measuring the time taken by clients to download/upload the file.

EXAMPLE-1:
Command Line Interface to run download scenario for Real clients
python3 lf_ftp.py --ssid Netgear-5g --passwd sharedsecret --file_sizes 10MB --mgr 192.168.200.165 --traffic_duration 1m
--security wpa2 --directions Download --clients_type Real --ap_name Netgear --bands 5G --upstream_port eth1

EXAMPLE-2:
Command Line Interface to run upload scenario on 6GHz band for Virtual clients
python3 lf_ftp.py --ssid Netgear-6g --passwd sharedsecret --file_sizes 10MB --mgr 192.168.200.165 --traffic_duration 1m
--security wpa3 --fiveg_radio wiphy2 --directions Upload --clients_type Virtual --ap_name Netgear --bands 6G --num_stations 2
--upstream_port eth1

EXAMPLE-3:
Command Line Interface to run download scenario on 5GHz band for Virtual clients
python3 lf_ftp.py --ssid Netgear-5g --passwd sharedsecret --file_sizes 10MB --mgr 192.168.200.165 --traffic_duration 1m
--security wpa2 --fiveg_radio wiphy2 --directions Download --clients_type Virtual --ap_name Netgear --bands 5G --num_stations 2
--upstream_port eth1

EXAMPLE-4:
Command Line Interface to run upload scenario for Real clients
python3 lf_ftp.py --ssid Netgear-2g --passwd sharedsecret --file_sizes 10MB --mgr 192.168.200.165 --traffic_duration 1m
--security wpa2 --directions Download --clients_type Real --ap_name Netgear --bands 2.4G --upstream_port eth1

EXAMPLE-5:
Command Line Interface to run upload scenario on 2.4GHz band for Virtual clients
python3 lf_ftp.py --ssid Netgear-2g --passwd sharedsecret --file_sizes 10MB --mgr 192.168.200.165 --traffic_duration 1m
--security wpa2 --twog_radio wiphy1 --directions Upload --clients_type Virtual --ap_name Netgear --bands 2.4G --num_stations 2
--upstream_port eth1

EXAMPLE-6:
Command Line Interface to run download scenario for Real clients with device list
python3 lf_ftp.py --ssid Netgear-2g --passwd sharedsecret --file_sizes 10MB --mgr 192.168.214.219 --traffic_duration 1m --security
--directions Download --clients_type Real --ap_name Netgear --bands 2.4G --upstream_port eth1 --device_list 1.12,1.22

SCRIPT_CLASSIFICATION : Test

SCRIPT_CATEGORIES:   Performance,  Functional,  Report Generation

NOTES:
After passing cli, a list will be displayed on terminal which contains available resources to run test.
The following sentence will be displayed
Enter the desired resources to run the test:
Please enter the port numbers seperated by commas ','.
Example:
Enter the desired resources to run the test:1.10,1.11,1.12,1.13,1.202,1.203,1.303

STATUS : Functional

VERIFIED_ON:
26-JULY-2024,
GUI Version:  5.4.8
Kernel Version: 6.2.16+

LICENSE :
Copyright 2023 Candela Technologies Inc
Free to distribute and modify. LANforge systems must be licensed.

INCLUDE_IN_README: False



optional arguments:
  -h, --help            show this help message and exit

Required arguments to run lf_ftp.py:
  --mgr MGR             hostname for where LANforge GUI is running [default = localhost]
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on [default = 8080]
  --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: eg: eth1 [default = eth1]
  --ssid SSID           Enter ssid
  --passwd PASSWD       Enter password for ssid provided
  --security SECURITY   Enter the security
  --ap_name AP_NAME     Enter the Access point or router name
  --traffic_duration TRAFFIC_DURATION
                        duration for layer 4 traffic running in minutes or seconds or hours. Example : 30s,3m,48h
  --clients_type CLIENTS_TYPE
                        Enter the type of clients on which the test is to be run. Example: "Virtual","Real"
  --dowebgui DOWEBGUI   If true will execute script for webgui
  --directions DIRECTIONS [DIRECTIONS ...]
                        Enter the traffic direction. Example : "Download","Upload"
  --file_sizes FILE_SIZES [FILE_SIZES ...]
                        File Size Example : "1000MB"

Optional arguments to run lf_ftp.py:
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path, primary use when running test in test suite
  --ap_ip AP_IP         Enter ip of accesspoint or router
  --twog_radio TWOG_RADIO
                        specify radio for 2.4G clients [default = wiphy1]
  --fiveg_radio FIVEG_RADIO
                        specify radio for 5G client [default = wiphy0]
  --sixg_radio SIXG_RADIO
                        specify radio for 6G clients [default = wiphy2]
  --lf_username LF_USERNAME
                        Enter the lanforge user name. Example : 'lanforge'
```

```
  --lf_password LF_PASSWORD
                        Enter the lanforge password. Example : 'lanforge'
  --ssh_port SSH_PORT   specify the shh port: eg 22 [default = 22]
  --bands BANDS [BANDS ...]
                        select bands for virtul clients Example : "5G","2.4G","6G"
  --num_stations NUM_STATIONS
                        number of virtual stations
  --result_dir RESULT_DIR
                        Specify the result dir to store the runtime logs
  --device_list DEVICE_LIST
                        Enter the devices on which the test should be run
  --test_name TEST_NAME
                        Specify test name to store the runtime csv results
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differenciate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number / serial number of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --test_id TEST_ID     test-id for kpi.csv,  script or test name
  --csv_outfile CSV_OUTFILE
                        --csv_outfile <Output file for csv data>
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --help_summary        Show summary of what this script does
```

py-scripts/lf_graph.py

```
usage: lf_graph.py [-h] [--mgr LFMGR] [--debug] [--log_level LOG_LEVEL]
                   [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                   [--help_summary]

-----------------
NAME: lf_graph.py

PURPOSE:
Common Library for generating graphs for LANforge output

SETUP:
/lanforge/html-reports directory needs to be present or output generated in local file

EXAMPLE:
see: /py-scritps/lf_report_test.py for example

COPYWRITE
    Copyright 2021 Candela Technologies Inc
    License: Free to distribute and modify. LANforge systems must be licensed.

INCLUDE_IN_README
---------------------


optional arguments:
  -h, --help            show this help message and exit
  --mgr LFMGR, --lfmgr LFMGR
                        sample argument: where LANforge GUI is running
  --debug               --debug this will enable debugging in py-json method
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --help_summary        Show summary of what this script does

           lf_graph.py : unit test in lf_graph.py for exersizing the lf_graph.py library
```

py-scripts/lf_interference.py

```
usage: lf_interference.py [-h] [-m MGR] [-o PORT] [--lf_user LF_USER]
                          [--lf_password LF_PASSWORD] [-i INSTANCE_NAME]
                          [-c CONFIG_NAME] [-r] [--load_old_cfg]
                          [--enable ENABLE] [--disable DISABLE]
                          [--set SET SET] [--raw_line RAW_LINE]
                          [--raw_lines_file RAW_LINES_FILE]
                          [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                          [-u DUT_UPSTREAM] [-b BATCH_SIZE] [-l LOOP_ITER]
                          [-p PROTOCOL] [-d DURATION]
                          [--download_rate DOWNLOAD_RATE]
                          [--upload_rate UPLOAD_RATE] [--sort SORT]
                          [-s STATIONS] [-cs] [-prf] [-radio RADIO]
                          [-ssid SSID] [-security SECURITY] [-paswd PASWD]
                          [--report_dir REPORT_DIR] [--scenario SCENARIO]
                          [--graph_groups GRAPH_GROUPS]
                          [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                          [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                          [-dos] [-sn SCENARIO_NAME] [-vr VAP_RADIO]
                          [-vf VAP_FREQ] [-vs VAP_SSID] [-vp VAP_PASSWD]
                          [-vse VAP_SECURITY] [-vsnc VAP_STA_NUMBER] [-vsue]
                          [-vsr VAP_STA_RADIO] [-vsname VAP_STA_NAME]
                          [-vstp VAP_STA_TRAFFIC_TYPE] [-vsipv6]
                          [-vsupstream VAP_STA_UPSTREAM_PORT]
                          [-vss VAP_STA_SSID] [-vsp VAP_STA_PASSWD]
                          [-vssec VAP_STA_SECURITY]
                          [-vstd VAP_STA_TEST_DURATION] [-vsam VAP_STA_A_MIN]
                          [-vsbm VAP_STA_B_MIN] [-vsmode VAP_STA_MODE]
                          [-vsap VAP_STA_AP] [-vsapc]
                          [-vsmi VAP_STA_MONITOR_INTERVAL]
                          [-vsl3c VAP_STA_LAYER3_COLS] [-vsmc PORT_MGR_COLS]
                          [-vsd] [--help_summary]

       ./lf_interference.py --mgr 192.168.200.38 --port 8080 --lf_user lanforge --lf_password lanforge
```

```
       --dut_upstream 1.1.eth1 --batch_size 1 --loop_iter 1 --protocol UDP-IPv4 --duration 240000 --download_rate 1Gbps
        --upload_rate 10Mbps --sort interleave --stations 1.1.sta0000 --create_stations --pull_report --radio "wiphy3"
        --ssid "NETGEAR-2G" --security "wpa2" --paswd "[BLANK]" --delete_old_scenario --scenario_name "Automation"
        --vap_radio "wiphy0" --vap_freq "2437" --vap_ssid "routed-AP" --vap_passwd "something" --vap_security "wpa2"
        --vap_sta_number 1 --vap_sta_radio "wiphy2" --vap_sta_name sta0000 --vap_sta_traffic_type lf_udp
        --vap_sta_upstream_port 1.1.vap0000 --vap_sta_ssid "routed-AP" --vap_sta_passwd "something"
        --vap_sta_security "wpa2" --vap_sta_test_duration "60s" --vap_sta_a_min 600000000 --vap_sta_b_min 600000000
        --vap_sta_mode 5 --vap_sta_cleanup --vap_sta_monitor_interval 10s


optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the capacity test?  Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).  See example raw text config for possible options.  M
  --disable DISABLE     Specify options to disable (set value to 0).  See example raw text config for possible options.  May be sp
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for instance:  testbed-01
  -u DUT_UPSTREAM, --dut_upstream DUT_UPSTREAM
                        Upstream port for wifi capacity test ex. 1.1.eth1
  -b BATCH_SIZE, --batch_size BATCH_SIZE
                        station increment ex. 1,2,3
  -l LOOP_ITER, --loop_iter LOOP_ITER
                        Loop iteration ex. 1
  -p PROTOCOL, --protocol PROTOCOL
                        Protocol ex.TCP-IPv4
  -d DURATION, --duration DURATION
                        duration in ms. ex. 5000
  --download_rate DOWNLOAD_RATE
                        Select requested download rate.  Kbps, Mbps, Gbps units supported.  Default is 1Gbps
  --upload_rate UPLOAD_RATE
                        Select requested upload rate.  Kbps, Mbps, Gbps units supported.  Default is 10Mbps
  --sort SORT           Select station sorting behaviour:  none | interleave | linear  Default is interleave.
  -s STATIONS, --stations STATIONS
                        If specified, these stations will be used.  If not specified, all available stations will be selected.  Ex
  -cs, --create_stations
                        create stations in lanforge (by default: False)
  -prf, --pull_report_flag
                        pull report from lanforge (by default: False)
  -radio RADIO, --radio RADIO
                        create stations in lanforge at this radio (by default: wiphy0)
  -ssid SSID, --ssid SSID
                        ssid name
  -security SECURITY, --security SECURITY
                        ssid Security type
  -paswd PASWD, --paswd PASWD
                        ssid Password
  --report_dir REPORT_DIR
  --scenario SCENARIO
  --graph_groups GRAPH_GROUPS
                        File to save graph groups to
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir <where to pull reports to>  default '' put where dataplane script run from
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  -dos, --delete_old_scenario
                        To delete old scenarios (by default: True)
  -sn SCENARIO_NAME, --scenario_name SCENARIO_NAME
                        Chamberview scenario name (by default: Automation
  -vr VAP_RADIO, --vap_radio VAP_RADIO
                        vap radio name (by default: wiphy0
  -vf VAP_FREQ, --vap_freq VAP_FREQ
                        vap frequency (by default: 2437
  -vs VAP_SSID, --vap_ssid VAP_SSID
                        vap ssid (by default: routed-AP
  -vp VAP_PASSWD, --vap_passwd VAP_PASSWD
                        vap password (by default: something
  -vse VAP_SECURITY, --vap_security VAP_SECURITY
                        vap security (by default: wpa2
  -vsnc VAP_STA_NUMBER, --vap_sta_number VAP_STA_NUMBER
                        To set vap station radio (by default: wiphy2)
  -vsue, --vap_sta_use_existing_sta
                        To delete old scenarios (by default: True)
  -vsr VAP_STA_RADIO, --vap_sta_radio VAP_STA_RADIO
                        To set vap station radio (by default: wiphy2)
  -vsname VAP_STA_NAME, --vap_sta_name VAP_STA_NAME
                        To set vap station name (by default: sta0000
  -vstp VAP_STA_TRAFFIC_TYPE, --vap_sta_traffic_type VAP_STA_TRAFFIC_TYPE
                        To set vap station traffic (by default: lf_udp
  -vsipv6, --vap_sta_ipv6
  -vsupstream VAP_STA_UPSTREAM_PORT, --vap_sta_upstream_port VAP_STA_UPSTREAM_PORT
  -vss VAP_STA_SSID, --vap_sta_ssid VAP_STA_SSID
                        vap ssid (by default: routed-AP
  -vsp VAP_STA_PASSWD, --vap_sta_passwd VAP_STA_PASSWD
                        vap password (by default: something
  -vssec VAP_STA_SECURITY, --vap_sta_security VAP_STA_SECURITY
                        vap security (by default: wpa2
  -vstd VAP_STA_TEST_DURATION, --vap_sta_test_duration VAP_STA_TEST_DURATION
                        vap security (by default: wpa2
  -vsam VAP_STA_A_MIN, --vap_sta_a_min VAP_STA_A_MIN
                        vap security (by default: wpa2
  -vsbm VAP_STA_B_MIN, --vap_sta_b_min VAP_STA_B_MIN
                        vap security (by default: wpa2
  -vsmode VAP_STA_MODE, --vap_sta_mode VAP_STA_MODE
```

```
                        vap security (by default: wpa2
  -vsap VAP_STA_AP, --vap_sta_ap VAP_STA_AP
  -vsapc, --vap_sta_cleanup
  -vsmi VAP_STA_MONITOR_INTERVAL, --vap_sta_monitor_interval VAP_STA_MONITOR_INTERVAL
  -vsl3c VAP_STA_LAYER3_COLS, --vap_sta_layer3_cols VAP_STA_LAYER3_COLS
  -vsmc PORT_MGR_COLS, --port_mgr_cols PORT_MGR_COLS
  -vsd, --vap_sta_debug
  --help_summary        Show summary of what this script does
```

py-scripts/lf_interop_modify.py

```
usage: ./lf_interop_modify.py [-h] [--debug] [--host HOST]
                              [--show_gui SHOW_GUI] [--install INSTALL]
                              [--install_g INSTALL_G] [--wifi WIFI] [--start]
                              [--stop] [--apply] [--mgr_ip MGR_IP]
                              [--user_name USER_NAME] [--ssid SSID]
                              [--crypt CRYPT] [--passwd PASSWD]
                              [--log_dur LOG_DUR] [--device DEVICE]
                              [--screensize SCREENSIZE]
                              [--log_destination LOG_DESTINATION]
                              [--log_level LOG_LEVEL] [--set_adb_user_name]
                              [--adb_username ADB_USERNAME] [--list_ntwk]
                              [--forget_netwrk] [--ntwk_id NTWK_ID]
                              [--help_summary]

modifies interop device
    Operations:
    *    Example of loading Interop GUI:
    lf_interop_modify.py --show_gui 192.168.100.202:1 --device 1.1.KEBE2021070849 --screensize 0.4
    *    Example of installing APK:
    lf_interop_modify.py --install_g interop-5.4.5.apk --device 1.1.KEBE2021070849
    *    Example of capturing logs
    lf_interop_modify.py --log_dur 5 --device 1.1.KEBE2021070849 --log_destination foo4.txt
    *    Examples of enabling/disabling wifi
    lf_interop_modify.py --wifi enable --device 1.1.KEBE2021070849
    lf_interop_modify.py --wifi disable --device 1.1.KEBE2021070849
    *    Examples of starting/stopping Interop app
    lf_interop_modify.py --start --device 1.1.KEBE2021070849
    lf_interop_modify.py --stop --device 1.1.KEBE2021070849
    *    Example of applying WiFi connection changes
    lf_interop_modify.py --apply --device 1.1.KEBE2021070849 --ssid candela-10g --user_name foobar22
    * Example for applying adb_username
    lf_interop_modify.py --host 192.168.1.31 --device 1.1.RZ8RA1053HJ --set_adb_user_name --adb_username device_1
    * Example for getting network-id list
    lf_interop_modify.py --host 192.168.1.31 --device 1.1.RZ8RA1053HJ --list_ntw
    * Example to forget a network
    lf_interop_modify.py --host 192.168.1.31 --device 1.1.RZ8RA1053HJ --ntwk_id 2  --forget_netwrk


optional arguments:
  -h, --help            show this help message and exit
  --debug               turn on debugging
  --host HOST, --mgr HOST
                        specify the GUI to connect to, assumes port 8080
  --show_gui SHOW_GUI, --gui SHOW_GUI
                        Display the Android GUI on this X-windows display address (IP:display). EG: 192.168.100.264:0.0
  --install INSTALL, --i INSTALL
                        Install apk with this filename
  --install_g INSTALL_G, --ig INSTALL_G
                        Install apk with this filename, adding the -g flag
  --wifi WIFI, --w WIFI
                        Enable or disable WiFi (enable | disable)
  --start               Start the LANforge Interop GUI
  --stop                Stop the LANforge Interop GUI
  --apply               Apply changes for (LF Mgr IP, Encryption, SSID, Passwd
  --mgr_ip MGR_IP       APPLY: IP address of the LF Manager managing this Interop device
  --user_name USER_NAME, --un USER_NAME
                        APPLY: Interop device user name
  --ssid SSID           APPLY: SSID for Interop device WiFi connection
  --crypt CRYPT, --enc CRYPT
                        APPLY: Encryption for Interop device WiFi connection
  --passwd PASSWD, --pw PASSWD
                        APPLY: Password for Interop device WiFi connection
  --log_dur LOG_DUR, --ld LOG_DUR
                        LOG: Gather ADB logs for a duration of this many minutes
  --device DEVICE, --dev DEVICE
                        specify the EID (serial number) of the interop device (eg 1.1.91BX93V4
  --screensize SCREENSIZE
                        GUI: specify the Android screen size when launching the Android GUI (percent as float)
  --log_destination LOG_DESTINATION, --log_dest LOG_DESTINATION
                        LOG: the filename destination on the LF device where the log file should be storedGive "stdout" to receive
  --log_level LOG_LEVEL
  --set_adb_user_name   provided when want to configure adb_username
  --adb_username ADB_USERNAME
                        provide user name to adb devices
  --list_ntwk           stores true when you want to get list of networks
  --forget_netwrk       stores true when you want to forget all wifi-networks
  --ntwk_id NTWK_ID     provide network id which you want to forget
  --help_summary        Show summary of what this script does
```

py-scripts/lf_interop_pdu_automation.py

```
usage: lf_interop_pdu_automation.py [-h] [--host HOST] [--username USERNAME]
                                    [--password PASSWORD] [--port PORT]
                                    [--current CURRENT] [--on_time ON_TIME]
                                    [--off_time OFF_TIME] [--help_summary]

optional arguments:
  -h, --help           show this help message and exit
  --host HOST          Please provide host name eg: 192.168.200.65
  --username USERNAME  Please provide username eg: admin
  --password PASSWORD  Please provide password eg: 1234
  --port PORT          Port number which has to be switched eg: --port 1,2,3,4
  --current CURRENT    Current status after running the code eg: --current
                       off/on
  --on_time ON_TIME    Time in (integer)minutes for keeping power on eg:
                       --on_time 2
```

```
  --off_time OFF_TIME  Time in (integer)minutes for keeping power off eg:
                       --off_time 5
  --help_summary       Show summary of what this script does
```

py-scripts/lf_interop_ping_plotter.py

```
usage: interop_ping.py [-h] [--mgr MGR] [--target TARGET]
                       [--ping_interval PING_INTERVAL]
                       [--ping_duration PING_DURATION] [--ssid SSID]
                       [--mgr_port MGR_PORT] [--mgr_passwd MGR_PASSWD]
                       [--server_ip SERVER_IP] [--security SECURITY]
                       [--passwd PASSWD] [--virtual] [--num_sta NUM_STA]
                       [--radio RADIO] [--real] [--use_default_config]
                       [--debug] [--no_cleanup] [--do_webUI]
                       [--resources RESOURCES] [--ui_report_dir UI_REPORT_DIR]
                       [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                       [--log_level LOG_LEVEL]
                       [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                       [--help_summary]

        NAME: lf_interop_ping_plotter.py

        PURPOSE: lf_interop_ping_plotter.py will let the user select real devices, virtual devices or both and then allows them to
        ping plotter test for user given duration and packet interval on the given target IP or domain name and generates realtime

        EXAMPLE-1:
        Command Line Interface to run ping plotter test with only virtual clients with eth1 as the default target
        python3 lf_interop_ping_plotter.py --mgr 192.168.200.103 --virtual --num_sta 1 --radio 1.1.wiphy2 --ssid RDT_wpa2 --securi
        --passwd OpenWifi --ping_interval 1 --ping_duration 1m --server_ip 192.168.1.61 --debug

        EXAMPLE-2:
        Command Line Interface to stop cleaning up stations after the test
        python3 lf_interop_ping_plotter.py --mgr 192.168.200.103 --virtual --num_sta 1 --radio 1.1.wiphy2 --ssid RDT_wpa2 --securi
        --passwd OpenWifi --ping_interval 1 --ping_duration 1m --server_ip 192.168.1.61 --debug --no_cleanup

        EXAMPLE-3:
        Command Line Interface to run ping plotter test with only real clients
        python3 lf_interop_ping_plotter.py --mgr 192.168.200.103 --real --ping_interval 1 --ping_duration 1m --server_ip 192.168.1

        EXAMPLE-4:
        Command Line Interface to run ping plotter test with both real and virtual clients
        python3 lf_interop_ping_plotter.py --mgr 192.168.200.103 --real --virtual --num_sta 1 --radio 1.1.wiphy2 --ssid RDT_wpa2 -
        --passwd OpenWifi --ping_interval 1 --ping_duration 1m --server_ip 192.168.1.61

        EXAMPLE-5:
        Command Line Interface to run ping plotter test with a different target
        python3 lf_interop_ping_plotter.py --mgr 192.168.200.63 --real --ping_interval 5 --ping_duration 1m --target 192.168.1.61

        SCRIPT_CLASSIFICATION : Test

        SCRIPT_CATEGORIES: Performance, Functional, Report Generation

        NOTES:
        1.Use './lf_interop_ping.py --help' to see command line usage and options
        2.Use 's','m','h' as suffixes for ping_duration in seconds, minutes and hours respectively
        3.After passing the cli, if --real flag is selected, then a list of available real devices will be displayed on the termin
        4.Enter the real device resource numbers seperated by commas (,)
        5.For --target, you can specify it as eth1, IP address or domain name (e.g., google.com)

        STATUS: BETA RELEASE

        VERIFIED_ON:
        Working date    - 06/12/2023
        Build version   - 5.4.7
        kernel version  - 6.2.16+

        License: Free to distribute and modify. LANforge systems must be licensed.
        Copyright 2023 Candela Technologies Inc.


optional arguments:
  -h, --help            show this help message and exit
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path (lanforge/html-reports), primary used when making another d
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --help_summary        Show summary of what this script does

Optional arguments:
  --mgr MGR             hostname where LANforge GUI is running
  --target TARGET       Target URL or port for ping plotter test
  --ping_interval PING_INTERVAL
                        Interval (in seconds) between the echo requests
  --ping_duration PING_DURATION
                        Duration to run the ping plotter test
  --ssid SSID           SSID for connecting the stations
  --mgr_port MGR_PORT   port on which LANforge HTTP service is running
  --mgr_passwd MGR_PASSWD
                        Password to connect to LANforge GUI
  --server_ip SERVER_IP
                        Upstream for configuring the Interop App
  --security SECURITY   Security protocol for the specified SSID: <open | wep | wpa | wpa2 | wpa3>
  --passwd PASSWD       passphrase for the specified SSID
  --virtual             specify this flag if the test should run on virtual clients
  --num_sta NUM_STA     specify the number of virtual stations to be created.
  --radio RADIO         specify the radio to create the virtual stations
  --real                specify this flag if the test should run on real clients
  --use_default_config  specify this flag if wanted to proceed with existing Wi-Fi configuration of the devices
  --debug               Enable debugging
  --no_cleanup          specify this flag to stop cleaning up generic cxs after the test

webUI arguments:
  --do_webUI            specify this flag when triggering a test from webUI
  --resources RESOURCES
                        Specify the real device ports seperated by comma
```

```
  --ui_report_dir UI_REPORT_DIR
                        Specify the results directory to store the reports for webUI

            Allows user to run the ping plotter test on a target IP for the given duration and packet interval
            with either selected number of virtual stations or provides the list of available real devices
            and allows the user to select the real devices and run ping plotter test on them.
```

py-scripts/lf_interop_ping.py

Example report: interop_ping.pdf

```
usage: interop_ping.py [-h] [--mgr MGR] [--target TARGET]
                       [--ping_interval PING_INTERVAL]
                       [--ping_duration PING_DURATION] [--ssid SSID]
                       [--mgr_port MGR_PORT] [--mgr_passwd MGR_PASSWD]
                       [--server_ip SERVER_IP] [--security SECURITY]
                       [--passwd PASSWD] [--virtual] [--num_sta NUM_STA]
                       [--radio RADIO] [--real] [--use_default_config]
                       [--debug] [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                       [--log_level LOG_LEVEL]
                       [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                       [--help_summary]

        NAME: lf_interop_ping.py

        PURPOSE: lf_interop_ping.py will let the user select real devices, virtual devices or both and then allows them to run
        ping test for user given duration and packet interval on the given target IP or domain name.

        EXAMPLE-1:
        Command Line Interface to run ping test with only virtual clients
        python3 lf_interop_ping.py --mgr 192.168.200.103  --target 192.168.1.3 --virtual --num_sta 1 --radio 1.1.wiphy2 --ssid RDT
        --passwd OpenWifi --ping_interval 1 --ping_duration 1 --server_ip 192.168.1.61 --debug

        EXAMPLE-2:
        Command Line Interface to run ping test with only real clients
        python3 lf_interop_ping.py --mgr 192.168.200.103 --real --target 192.168.1.3 --ping_interval 1 --ping_duration 1 --server_
        --security wpa2_personal --passwd OpenWifi

        EXAMPLE-3:
        Command Line Interface to run ping test with both real and virtual clients
        python3 lf_interop_ping.py --mgr 192.168.200.103 --target 192.168.1.3 --real --virtual --num_sta 1 --radio 1.1.wiphy2 --ss
        --passwd OpenWifi --ping_interval 1 --ping_duration 1 --server_ip 192.168.1.61

        EXAMPLE-4:
        Command Line Interface to run ping test with existing Wi-Fi configuration on the real devices
        python3 lf_interop_ping.py --mgr 192.168.200.63 --real --target 192.168.1.61 --ping_interval 5 --ping_duration 1 --passwd

        SCRIPT_CLASSIFICATION : Test

        SCRIPT_CATEGORIES: Performance, Functional, Report Generation

        NOTES:
        1.Use './lf_interop_ping.py --help' to see command line usage and options
        2.Please pass ping_duration in minutes
        3.Please pass ping_interval in seconds
        4.After passing the cli, if --real flag is selected, then a list of available real devices will be displayed on the termin
        5.Enter the real device resource numbers seperated by commas (,)

        STATUS: BETA RELEASE

        VERIFIED_ON:
        Working date    - 20/09/2023
        Build version   - 5.4.7
        kernel version  - 6.2.16+

        License: Free to distribute and modify. LANforge systems must be licensed.
        Copyright 2023 Candela Technologies Inc.


optional arguments:
  -h, --help            show this help message and exit
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path (lanforge/html-reports), primary used when making another d
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --help_summary        Show summary of what this script does

Optional arguments:
  --mgr MGR             hostname where LANforge GUI is running
  --target TARGET       Target URL or port for ping test
  --ping_interval PING_INTERVAL
                        Interval (in seconds) between the echo requests
  --ping_duration PING_DURATION
                        Duration (in minutes) to run the ping test
  --ssid SSID           SSID for connecting the stations
  --mgr_port MGR_PORT   port on which LANforge HTTP service is running
  --mgr_passwd MGR_PASSWD
                        Password to connect to LANforge GUI
  --server_ip SERVER_IP
                        Upstream for configuring the Interop App
  --security SECURITY   Security protocol for the specified SSID: <open | wep | wpa | wpa2 | wpa3>
  --passwd PASSWD       passphrase for the specified SSID
  --virtual             specify this flag if the test should run on virtual clients
  --num_sta NUM_STA     specify the number of virtual stations to be created.
  --radio RADIO         specify the radio to create the virtual stations
  --real                specify this flag if the test should run on real clients
  --use_default_config  specify this flag if wanted to proceed with existing Wi-Fi configuration of the devices
  --debug               Enable debugging

            Allows user to run the ping test on a target IP or port for the given duration and packet interval
            with either selected number of virtual stations or provides the list of available real devices
            and allows the user to select the real devices and run ping test on them.
```

py-scripts/lf_interop_port_reset_test.py

```
usage: ./lf_interop_port_reset_test.py [-h] [--host HOST] [--port PORT]
                                       [--mgr_ip MGR_IP] [--dut DUT]
                                       [--ssid SSID] [--passwd PASSWD]
                                       [--encryp ENCRYP] [--reset RESET]
                                       [--time_int TIME_INT]
                                       [--release RELEASE [RELEASE ...]]
                                       [--log_level LOG_LEVEL]
                                       [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                       [--help_summary]

NAME: lf_interop_port_reset_test.py

PURPOSE:
         The LANforge interop port reset test enables users to use real Wi-Fi stations and connect them to the
         Access Point (AP) being tested. It then disconnects and reconnects a given number of stations at
         different time intervals. This test helps evaluate how well the AP handles a dynamic and busy network environment
         with devices joining and leaving the network at random times.

EXAMPLE:
        # To run port-reset test on specified real devices (android, laptops)

            python3 lf_interop_port_reset_test.py --host 192.168.200.63 --mgr_ip 192.168.1.61 --dut Test_Dut
            --ssid RDT_wpa2 --passwd OpenWifi --encryp psk2 --reset 1 --time_int 5 --release 11

SCRIPT_CLASSIFICATION:  Interop Port-Reset Test

SCRIPT_CATEGORIES: Toggling, Report Generation, Each Reset Wifi Messages

NOTES:
        The primary objective of this script is to automate the process of toggling WiFi on real devices with the
       InterOp Application, evaluating their performance with an access point. It achieves this by simulating multiple
       WiFi resets as specified by the user.

      * Currently the script will work for the REAL CLIENTS (android with version 11+, laptop devices).

STATUS: Functional

VERIFIED_ON:   28-OCT-2023,
             GUI Version:  5.4.7
             Kernel Version: 6.2.16+

LICENSE:
          Free to distribute and modify. LANforge systems must be licensed.
          Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False

optional arguments:
  -h, --help            show this help message and exit
  --host HOST           Specify the GUI to connect to, assumes port 8080
  --port PORT           Specify the manager port
  --mgr_ip MGR_IP       Specify the interop manager ip
  --dut DUT             Specify DUT name on which the test will be running.
  --ssid SSID           Specify ssid on which the test will be running.
  --passwd PASSWD       Specify encryption password  on which the test will be running.
  --encryp ENCRYP       Specify the encryption type  on which the test will be running eg :open|psk|psk2|sae|psk2jsae
  --reset RESET         Specify the number of time you want to reset. eg: 2
  --time_int TIME_INT   Specify the time interval in seconds after which reset should happen.
  --release RELEASE [RELEASE ...]
                        Specify the SDK release version (Android Version) of real clients to be supported in test.eg:- --release 1
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --help_summary        Show summary of what this script does
```

py-scripts/lf_interop_qos.py

```
usage: throughput_QOS.py [-h] [--device_list DEVICE_LIST]
                         [--test_name TEST_NAME] [--result_dir RESULT_DIR]
                         [--mgr MGR] [--mgr_port MGR_PORT]
                         [--upstream_port UPSTREAM_PORT] [--security SECURITY]
                         [--ssid SSID] [--passwd PASSWD]
                         [--traffic_type TRAFFIC_TYPE] [--upload UPLOAD]
                         [--download DOWNLOAD] [--test_duration TEST_DURATION]
                         [--ap_name AP_NAME] [--tos TOS] [--dowebgui DOWEBGUI]
                         [-d] [--help_summary]


        NAME: lf_interop_qos.py

        PURPOSE: lf_interop_qos.py will provide the available devices and allows user to run the qos traffic
        with particular tos on particular devices in upload, download directions.

        EXAMPLE-1:
        Command Line Interface to run download scenario with tos : Voice
        python3 lf_interop_qos.py --ap_name Cisco --mgr 192.168.209.223 --mgr_port 8080 --ssid Cisco
        --passwd cisco@123 --security wpa2 --upstream eth1 --test_duration 1m --download 1000000 --upload 0
        --traffic_type lf_udp --tos "VO"

        EXAMPLE-2:
        Command Line Interface to run download scenario with tos : Voice and Video
        python3 lf_interop_qos.py --ap_name Cisco --mgr 192.168.209.223 --mgr_port 8080 --ssid Cisco
        --passwd cisco@123 --security wpa2 --upstream eth1 --test_duration 1m --download 1000000 --upload 0
        --traffic_type lf_udp --tos "VO,VI"

        EXAMPLE-3:
        Command Line Interface to run upload scenario with tos : Background, Besteffort, Video and Voice
        python3 lf_interop_qos.py --ap_name Cisco --mgr 192.168.209.223 --mgr_port 8080 --ssid Cisco
        --passwd cisco@123 --security wpa2 --upstream eth1 --test_duration 1m --download 0 --upload 1000000
        --traffic_type lf_udp --tos "BK,BE,VI,VO"

        EXAMPLE-4:
        Command Line Interface to run bi-directional scenario with tos : Video and Voice
        python3 lf_interop_qos.py --ap_name Cisco --mgr 192.168.209.223 --mgr_port 8080 --ssid Cisco
        --passwd cisco@123 --security wpa2 --upstream eth1 --test_duration 1m --download 1000000 --upload 1000000
        --traffic_type lf_udp --tos "VI,VO"
```

```
        SCRIPT_CLASSIFICATION :  Test

        SCRIPT_CATEGORIES:   Performance,  Functional, Report Generation

        NOTES:
        1.Use './lf_interop_qos.py --help' to see command line usage and options
        2.Please pass tos in CAPITALS as shown :"BK,VI,BE,VO"
        3.Please enter the download or upload rate in bps
        4.After passing cli, a list will be displayed on terminal which contains available resources to run test.
        The following sentence will be displayed
        Enter the desired resources to run the test:
        Please enter the port numbers seperated by commas ','.
        Example:
        Enter the desired resources to run the test:1.10,1.11,1.12,1.13,1.202,1.203,1.303

        STATUS: BETA RELEASE

        VERIFIED_ON:
        Working date - 26/07/2023
        Build version - 5.4.8
        kernel version - 6.2.16+

        License: Free to distribute and modify. LANforge systems must be licensed.
        Copyright 2023 Candela Technologies Inc.

optional arguments:
  -h, --help            show this help message and exit
  --help_summary        Show summary of what this script does

Required arguments to run lf_interop_qos.py:
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  --upstream_port UPSTREAM_PORT, -u UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key
  --traffic_type TRAFFIC_TYPE
                        Select the Traffic Type [lf_udp, lf_tcp]
  --upload UPLOAD       --upload traffic load per connection (upload rate)
  --download DOWNLOAD   --download traffic load per connection (download rate)
  --test_duration TEST_DURATION
                        --test_duration sets the duration of the test
  --ap_name AP_NAME     AP Model Name
  --tos TOS             Enter the tos. Example1 : "BK,BE,VI,VO" , Example2 : "BK,VO", Example3 : "VI"
  --dowebgui DOWEBGUI   If true will execute script for webgui

Optional arguments to run lf_interop_qos.py:
  --device_list DEVICE_LIST
                        Enter the devices on which the test should be run
  --test_name TEST_NAME
                        Specify test name to store the runtime csv results
  --result_dir RESULT_DIR
                        Specify the result dir to store the runtime logs
  -d, --debug           Enable debugging

            Provides the available devices list and allows user to run the qos traffic
            with particular tos on particular devices in upload, download directions.
```

py-scripts/lf_interop_real_browser_test.py

```
usage: ./lf_interop_real_browser_test.py [-h] [--host HOST] [--ssid SSID]
                                         [--passwd PASSWD] [--encryp ENCRYP]
                                         [--url URL] [--max_speed MAX_SPEED]
                                         [--urls_per_tenm URLS_PER_TENM]
                                         [--duration DURATION]
                                         [--test_name TEST_NAME]
                                         [--dowebgui DOWEBGUI]
                                         [--result_dir RESULT_DIR]
                                         [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                         [--log_level LOG_LEVEL] [--debug]
                                         [--device_list DEVICE_LIST]
                                         [--webgui_incremental WEBGUI_INCREMENTAL]
                                         [--incremental] [--no_laptops]
                                         [--postcleanup] [--precleanup]
                                         [--help_summary]


        Name: lf_interop_real_browser_test.py

        Purpose: To be generic script for LANforge-Interop devices(Real clients) which runs layer4-7 traffic
        For now the test script supports Real Browser test for Androids.

        Pre-requisites: Real devices should be connected to the LANforge MGR and Interop app should be open on the real clients wh

        Example: (python3 or ./)lf_interop_real_browser_test.py --mgr 192.168.214.219 --duration 1 --url "www.google.com"

        Example-1 :
        Command Line Interface to run url in the Browser with specified URL and duration:
        python3 lf_interop_real_browser_test.py --mgr 192.168.214.219 --url "www.google.com" --duration 10m --debug

            CASE-1:
            If not specified it takes the default url (default url is www.google.com)

        Example-2:
        Command Line Interface to run url in the Browser with specified Resources:
        python3 lf_interop_real_browser_test.py --mgr 192.168.214.219 --url "www.google.com" --duration 10m --device_list 1.10,1.1

        Example-3:
        Command Line Interface to run url in the Browser with specified urls_per_tennm (specify the number of url you want to test
        python3 lf_interop_real_browser_test.py --mgr 192.168.214.219 --url "www.google.com" --duration 10m --device_list 1.10,1.1
```

```
            CASE-1:
            If not specified it takes the default urls_per_tenm value (default urls_per_tenm is 100)

        Example-4:
        Command Line Interface to run the the Real Browser test with incremental Capacity by specifying the --incremental flag
        python3 lf_interop_real_browser_test.py --mgr 192.168.214.219 --url "www.google.com" --duration 10m --device_list 1.10,1.1

        Example-5:
        Command Line Interface to run the the Real Browser test in webGUI by specifying the --dowebgui flag
        python3 lf_interop_real_browser_test.py --mgr 192.168.214.219 --url "www.google.com" --duration 10m --device_list 1.10,1.1

        Example-6:
        Command Line Interface to run url in the Browser with precleanup:
        python3 lf_interop_real_browser_test.py --mgr 192.168.214.219 --url "www.google.com" --duration 10m --device_list 1.10,1.1

        Example-7:
        Command Line Interface to run url in the Browser with postcleanup:
        python3 lf_interop_real_browser_test.py --mgr 192.168.214.219 --url "www.google.com" --duration 10m --device_list 1.10,1.1

        SCRIPT CLASSIFICATION: Test

        SCRIPT_CATEGORIES:   Performance,  Functional, Report Generation

        NOTES:
            1. Use './lf_interop_real_browser_test.py --help' to see command line usage and options.
            2. Always specify the duration in minutes (for example: --duration 3 indicates a duration of 3 minutes).
            3. If --device_list are not given after passing the CLI, a list of available devices will be displayed on the terminal
            4. Enter the resource numbers separated by commas (,) in the resource argument and also enclose in double quotes (e.g.
            5. For --url, you can specify the URL (e.g., www.google.com).
            6. To run the test by specifying the incremental capacity, enable the --incremental flag.

        STATUS: BETA RELEASE

        VERIFIED_ON:
        Working date - 29/07/2024
        Build version - 5.4.8
        kernel version - 6.2.16+

        License: Free to distribute and modify. LANforge systems must be licensed.
        Copyright 2023 Candela Technologies Inc.



optional arguments:
  -h, --help            show this help message and exit
  --host HOST, --mgr HOST
                        specify the GUI to connect to, assumes port 8080
  --ssid SSID           specify ssid on which the test will be running
  --passwd PASSWD       specify encryption password  on which the test will be running
  --encryp ENCRYP       specify the encryption type  on which the test will be running eg :open|psk|psk2|sae|psk2jsae
  --url URL             specify the url you want to test on
  --max_speed MAX_SPEED
                        specify the max speed you want in bytes
  --urls_per_tenm URLS_PER_TENM
                        specify the number of url you want to test on per minute
  --duration DURATION   time to run traffic
  --dowebgui DOWEBGUI   If true will execute script for webgui
  --result_dir RESULT_DIR
                        Specify the result dir to store the runtime logs <Do not use in CLI, --used by webui>
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        [log configuration] --lf_logger_config_json <json file> , json configuration of logger
  --log_level LOG_LEVEL
                        [log configuration] --log_level  debug info warning error critical
  --debug               [log configuration] --debug store_true , used by lanforge client
  --device_list DEVICE_LIST
                        provide resource_ids of android devices. for instance: "10,12,14"
  --webgui_incremental WEBGUI_INCREMENTAL
                        Specify the incremental values <1,2,3..>
  --incremental         to add incremental capacity to run the test
  --postcleanup         Cleanup the cross connections after test is stopped
  --precleanup          Cleanup the cross connections before test is started
  --help_summary        Show summary of what this script does

Optional arguments to run lf_interop_real_browser_test.py:
  --test_name TEST_NAME
                        Specify test name to store the runtime csv results
  --no_laptops          run the test without laptop devices
```

py-scripts/lf_interop_rvr_test.py

```
usage: lf_interop_rvr_test.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                              [--upstream UPSTREAM] [--mode MODE]
                              [--ssid SSID] [--security SECURITY]
                              [--password PASSWORD]
                              [--traffic_type TRAFFIC_TYPE]
                              [--traffic_direction TRAFFIC_DIRECTION]
                              [--traffic TRAFFIC]
                              [--test_duration TEST_DURATION]
                              [--create_sta CREATE_STA]
                              [--sta_names STA_NAMES] [--ap_model AP_MODEL]
                              [-as ATTEN_SERNO] [-ai ATTEN_IDX]
                              [-av ATTEN_VAL] [--debug DEBUG]
                              [--log_level LOG_LEVEL]
                              [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                              [--help_summary]

rvr_test.py: -------------------- Generic command layout:
======================================================================== sudo
python3 rvr_test.py --mgr localhost --mgr_port 8080 --upstream eth1
--num_stations 40 --security wpa2 --ssid NETGEAR73-5G --password fancylotus986
--radio wiphy3 --atten_serno 2222 --atten_idx all --atten_val 10
--test_duration 1m --ap_model WAX610 --traffic 100

optional arguments:
  -h, --help            show this help message and exit
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error |
```

```
                        critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json
                        configuration of logger
  --help_summary        Show summary of what this script does

optional arguments:
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --upstream UPSTREAM   non-station port that generates traffic:
                        <resource>.<port>, e.g: 1.eth1
  --mode MODE           used to force mode of stations
  --traffic_direction TRAFFIC_DIRECTION
                        Traffic direction i.e upload or download or
                        bidirectional
  --create_sta CREATE_STA
                        used to create stations if you do not prefer existing
                        stations
  --sta_names STA_NAMES
                        used to provide existing station names from the port
                        manager, prefer only if create_sta is False
  --ap_model AP_MODEL   AP Model Name
  -as ATTEN_SERNO, --atten_serno ATTEN_SERNO
                        Serial number for requested Attenuator
  -ai ATTEN_IDX, --atten_idx ATTEN_IDX
                        Attenuator index eg. For module 1 = 0,module 2 = 1 -->
                        --atten_idx 0,1
  -av ATTEN_VAL, --atten_val ATTEN_VAL
                        Requested attenuation in dB ex:--> --atten_val
                        0..10..40 (here attenuation start from 0 and end with
                        50 with increment value of 10 each time)
  --debug DEBUG         to enable debug

required arguments:
  --ssid SSID           ssid for client association with Access Point,
                        REQUIRED
  --security SECURITY   security type of ssid, ex: wpa || wpa2 || wpa3 || open
                        REQUIRED
  --password PASSWORD   password of ssid
  --traffic_type TRAFFIC_TYPE
                        provide the traffic Type lf_udp, lf_tcp
  --traffic TRAFFIC     traffic to be created for the given number of clients
                        (in Mbps) REQUIRED
  --test_duration TEST_DURATION
                        sets the duration of the test ex: 2s --> two seconds
                        || 2m --> two minutes || 2h --> two hours Required
```

py-scripts/lf_interop_throughput.py

```
usage: lf_interop_throughputput.py [-h] [--device_list DEVICE_LIST]
                                   [--mgr MGR] [--mgr_port MGR_PORT]
                                   [--upstream_port UPSTREAM_PORT]
                                   [--ssid SSID] [--passwd PASSWD]
                                   [--traffic_type TRAFFIC_TYPE]
                                   [--upload UPLOAD] [--download DOWNLOAD]
                                   [--test_duration TEST_DURATION]
                                   [--report_timer REPORT_TIMER]
                                   [--ap_name AP_NAME] [--dowebgui]
                                   [--tos TOS] [--packet_size PACKET_SIZE]
                                   [--incremental_capacity INCREMENTAL_CAPACITY]
                                   [--load_type LOAD_TYPE]
                                   [--do_interopability] [--postcleanup]
                                   [--precleanup] [--incremental]
                                   [--security SECURITY]
                                   [--test_name TEST_NAME]
                                   [--result_dir RESULT_DIR] [--help_summary]


NAME: lf_interop_throughputput.py

PURPOSE: lf_interop_throughput.py will provide the available devices and allows user to run the wifi capacity test
on particular devices by specifying direction as upload, download and bidirectional including different types of loads and increme
Will also run the interopability test on particular devices by specifying direction as upload, download and bidirectional.

TO PERFORM THROUGHPUT TEST:

EXAMPLE-1:
Command Line Interface to run download scenario with desired resources
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080 --upstream_port eth1 --test_duration 1m --download 1000000

EXAMPLE-2:
Command Line Interface to run download scenario with incremental capacity
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080 --security wpa2 --upstream_port eth1 --test_duration 1m --d
--traffic_type lf_udp --incremental_capacity 1,2

EXAMPLE-3:
Command Line Interface to run upload scenario with packet size
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080 --security wpa2 --upstream_port eth1 --test_duration 1m --d

EXAMPLE-4:
Command Line Interface to run bi-directional scenario with load_type
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080 --security wpa2 --upstream_port eth1 --test_duration 1m --d
--traffic_type lf_udp --load_type wc_intended_load

EXAMPLE-5:
Command Line Interface to run bi-directional scenario with report_timer
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080 --security wpa2 --upstream_port eth1 --test_duration 1m --d
--traffic_type lf_udp --report_timer 5s

EXAMPLE-6:
Command Line Interface to run bi-directional scenario in Interop web-GUI
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080 --security wpa2 --upstream_port eth1 --test_duration 1m --d
--traffic_type lf_udp --report_timer 5s --dowebgui

EXAMPLE-7:
Command Line Interface to run the test with precleanup
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080  --upstream_port eth1 --test_duration 1m --download 1000000
```

```
EXAMPLE-8:
Command Line Interface to run the test with postcleanup
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080  --upstream_port eth1 --test_duration 1m --download 1000000

EXAMPLE-9:
Command Line Interface to run the test with incremental_capacity by raising incremental flag
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080  --upstream_port eth1 --test_duration 1m --download 1000000

TO PERFORM INTEROPABILITY TEST:

EXAMPLE-1:
Command Line Interface to run download scenario with desired resources
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080  --upstream_port eth1 --test_duration 1m --download 1000000

EXAMPLE-2:
Command Line Interface to run bi-directional scenario in Interop web-GUI
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080 --security wpa2 --upstream_port eth1 --test_duration 1m --d
--traffic_type lf_udp --do_interopability --dowebgui

EXAMPLE-3:
Command Line Interface to run the test with precleanup
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080  --upstream_port eth1 --test_duration 1m --download 1000000

EXAMPLE-4:
Command Line Interface to run the test with postcleanup
python3 lf_interop_throughput.py --mgr 192.168.214.219 --mgr_port 8080  --upstream_port eth1 --test_duration 1m --download 1000000
SCRIPT_CLASSIFICATION :  Test

SCRIPT_CATEGORIES:   Performance,  Functional, Report Generation

NOTES:
1.Use './lf_interop_qos.py --help' to see command line usage and options
2.Please enter the download or upload rate in bps
3.Inorder to perform intended load please pass 'wc_intended_load' in load_type argument.
4.Please pass incremental values seperated by commas ',' in incremental_capacity argument.
5.Please enter packet_size in bps.
6.After passing cli, a list will be displayed on terminal which contains available resources to run test.
The following sentence will be displayed
Enter the desired resources to run the test:
Please enter the port numbers seperated by commas ','.
Example:
Enter the desired resources to run the test:1.10,1.11,1.12,1.13,1.202,1.203,1.303

STATUS: BETA RELEASE

VERIFIED_ON:
Working date - 26/07/2024
Build version - 5.4.8
kernel version - 6.2.16+

License: Free to distribute and modify. LANforge systems must be licensed.
Copyright 2023 Candela Technologies Inc.

optional arguments:
  -h, --help            show this help message and exit
  --help_summary        Show summary of what this script does

Required arguments to run throughput.py:
  --device_list DEVICE_LIST
                        Enter the devices on which the test should be run
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  --upstream_port UPSTREAM_PORT, -u UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key
  --traffic_type TRAFFIC_TYPE
                        Select the Traffic Type [lf_udp, lf_tcp]
  --upload UPLOAD       --upload traffic load per connection (upload rate)
  --download DOWNLOAD   --download traffic load per connection (download rate)
  --test_duration TEST_DURATION
                        --test_duration sets the duration of the test
  --report_timer REPORT_TIMER
                        --duration to collect data
  --ap_name AP_NAME     AP Model Name
  --dowebgui            If true will execute script for webgui
  --tos TOS
  --packet_size PACKET_SIZE
                        Determine the size of the packet in which Packet Size Should be Greater than 16B or less than 64KB(65507)
  --incremental_capacity INCREMENTAL_CAPACITY
                        Specify the incremental values for network load testing as a comma-separated list (e.g., 10,20,30). This d
  --load_type LOAD_TYPE
                        Determine the type of load: < wc_intended_load | wc_per_client_load >
  --do_interopability

Optional arguments to run throughput.py:
  --postcleanup         Cleanup the cross connections after test is stopped
  --precleanup          Cleanup the cross connections before test is started
  --incremental         gives an option to the user to enter incremental values
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --test_name TEST_NAME
                        Specify test name to store the runtime csv results
  --result_dir RESULT_DIR
                        Specify the result dir to store the runtime logs

           Provides the available devices and allows user to run the wifi capacity test
           on particular devices by specifying direction as upload, download and bidirectional including different types of loads
```

py-scripts/lf_interop_video_streaming.py

```
usage: ./lf_interop_video_streaming.py [-h] [--host HOST] [--ssid SSID]
                                       [--passwd PASSWD] [--encryp ENCRYP]
```

```
                                        [--url URL] [--max_speed MAX_SPEED]
                                        [--urls_per_tenm URLS_PER_TENM]
                                        [--duration DURATION]
                                        [--test_name TEST_NAME]
                                        [--dowebgui DOWEBGUI]
                                        [--result_dir RESULT_DIR]
                                        [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                        [--log_level LOG_LEVEL] [--debug]
                                        [--media_source MEDIA_SOURCE]
                                        [--media_quality MEDIA_QUALITY]
                                        [--device_list DEVICE_LIST]
                                        [--webgui_incremental WEBGUI_INCREMENTAL]
                                        [--incremental] [--no_laptops]
                                        [--postcleanup] [--precleanup]
                                        [--help_summary]


        NAME: lf_interop_video_streaming.py

        Purpose: To be generic script for LANforge-Interop devices(Real clients) which runs layer4-7 traffic
        For now the test script supports for Video streaming of real devices.

        Pre-requisites: Real devices should be connected to the LANforge MGR and Interop app should be open on the real clients wh

        Prints the list of data from layer4-7 such as uc-avg time, total url's, url's per sec

        Example-1:
        Command Line Interface to run Video Streaming test with media source HLS and media quality 1080P :
        python3 lf_interop_video_streaming.py --mgr 192.168.214.219 --url "https://test-streams.mux.dev/x36xhzz/x36xhzz.m3u8" --me

        Example-2:
        Command Line Interface to run Video Streaming test with media source DASH and media quality 4K :
        python3 lf_interop_video_streaming.py --mgr 192.168.214.219 --url "https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps.m

        Example-3:
        Command Line Interface to run the Video Streaming test with specified Resources:
        python3 lf_interop_video_streaming.py --mgr 192.168.214.219 --url "https://test-streams.mux.dev/x36xhzz/x36xhzz.m3u8" --me

        Example-4:
        Command Line Interface to run the Video Streaming test with incremental Capacity by specifying the --incremental flag
        python3 lf_interop_video_streaming.py --mgr 192.168.214.219 --url "https://test-streams.mux.dev/x36xhzz/x36xhzz.m3u8" --me

        Example-5:
        Command Line Interface to run Video Streaming test with precleanup:
        python3 lf_interop_video_streaming.py --mgr 192.168.214.219 --url "https://test-streams.mux.dev/x36xhzz/x36xhzz.m3u8" --me

        Example-6:
        Command Line Interface to run Video Streaming test with postcleanup:
        python3 lf_interop_video_streaming.py --mgr 192.168.214.219 --url "https://test-streams.mux.dev/x36xhzz/x36xhzz.m3u8" --me

        SCRIPT CLASSIFICATION: Test

        SCRIPT_CATEGORIES:   Performance,  Functional, Report Generation

        NOTES:
            1. Use './lf_interop_video_streaming.py --help' to see command line usage and options.
            2. If --device_list are not given after passing the CLI, a list of available devices will be displayed on the terminal
            3. To run the test by specifying the incremental capacity, enable the --incremental flag.

        STATUS: BETA RELEASE

        VERIFIED_ON:
        Working date - 29/07/2024
        Build version - 5.4.8
        kernel version - 6.2.16+

        License: Free to distribute and modify. LANforge systems must be licensed.
        Copyright 2023 Candela Technologies Inc.



optional arguments:
  -h, --help            show this help message and exit
  --host HOST, --mgr HOST
                        specify the GUI to connect to, assumes port 8080
  --ssid SSID           specify ssid on which the test will be running
  --passwd PASSWD       specify encryption password  on which the test will be running
  --encryp ENCRYP       specify the encryption type  on which the test will be running eg :open|psk|psk2|sae|psk2jsae
  --url URL             specify the url you want to test on
  --max_speed MAX_SPEED
                        specify the max speed you want in bytes
  --urls_per_tenm URLS_PER_TENM
                        specify the number of url you want to test on per minute
  --duration DURATION   time to run traffic
  --test_name TEST_NAME
                        Name of the Test
  --dowebgui DOWEBGUI   If true will execute script for webgui
  --result_dir RESULT_DIR
                        Specify the result dir to store the runtime logs <Do not use in CLI, --used by webui>
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        [log configuration] --lf_logger_config_json <json file> , json configuration of logger
  --log_level LOG_LEVEL
                        [log configuration] --log_level  debug info warning error critical
  --debug               [log configuration] --debug store_true , used by lanforge client
  --media_source MEDIA_SOURCE
  --media_quality MEDIA_QUALITY
  --device_list DEVICE_LIST
                        provide resource_ids of android devices. for instance: "10,12,14"
  --webgui_incremental WEBGUI_INCREMENTAL
                        Specify the incremental values <1,2,3..>
  --incremental         --incremental to add incremental values
  --no_laptops          --to not use laptops
  --postcleanup         Cleanup the cross connections after test is stopped
  --precleanup          Cleanup the cross connections before test is started
  --help_summary        Show summary of what this script does
```

```
usage: lf_json_api.py [-h] [--lf_mgr LF_MGR] [--lf_port LF_PORT]
                      [--lf_user LF_USER] [--lf_passwd LF_PASSWD]
                      [--port PORT] [--endpoint ENDPOINT] [--radio RADIO]
                      [--log_level LOG_LEVEL]
                      [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                      [--get_requests GET_REQUESTS]
                      [--post_requests POST_REQUESTS] [--nss NSS]
                      [--csv_mode {append,write}] [--help_summary]

        The script will read column data from lanforge GUI using request

    EXAMPLE:

    ./lf_json_api.py --lf_mgr 192.168.100.116 --lf_port 8080 --log_level debug --port wlan3 --lf_user lanforge --lf_passwd lanforg
        --port 1.1.vap0000 --get_request 'stations,04:f0:21:c5:33:97 stations,d8:f8:83:36:6c:44'

    ./lf_json_api.py --lf_mgr 192.168.100.116 --lf_port 8080 --log_level debug --port wlan3 --lf_user lanforge --lf_passwd lanforg
        --port 1.1.vap0000 --get_request 'wifi-stats'

    # retrieve information form generic tab , generic tab need to be enabled
    ./lf_json_api.py --lf_mgr 192.168.100.116 --lf_port 8080 --log_level info --endpoint 1.1.21.5386 --lf_user lanforge --lf_passw

        This will generate a csv file generic_<endpoint>.csv



optional arguments:
  -h, --help            show this help message and exit
  --lf_mgr LF_MGR       address of the LANforge GUI machine (localhost is default)
  --lf_port LF_PORT     IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     user: lanforge
  --lf_passwd LF_PASSWD
                        passwd: lanforge
  --port PORT            port : 1.2.wlan3  provide full eid  (endpoint id
  --endpoint ENDPOINT    endpoint : shelf.resource.port.endpoint, provide full eid: 1.1.9.1178  (endpoint id)
  --radio RADIO          --radio wiphy0
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --get_requests GET_REQUESTS
                        perform get request may be a list:  port | radio | port_rssi | wifi-stats | stations | adb
  --post_requests POST_REQUESTS
                        perform set request may be a list:  nss , in development
  --nss NSS             --nss 4  set the number of spatial streams for a speific antenna
  --csv_mode {append,write}
                        --csv_mode 'write' or 'append' default: write
  --help_summary        Show summary of what this script does
```

py-scripts/lf_json_util.py

```
usage: lf_json_api.py [-h] [--help_summary]

This file contains a helper module standardize_json_results.
standardize_json_results takes a dict of information retrieved from json_get and standardizes
it to use the plural version of the data requested.
The data is returned starting with the "endpoints"
The script will read column data from lanforge GUI using request

optional arguments:
  -h, --help      show this help message and exit
  --help_summary  Show summary of what this script does
```

py-scripts/lf_kpi_csv.py

```
usage: lf_kpi_csv.py [-h] [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                     [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                     [--dut_hw_version DUT_HW_VERSION]
                     [--dut_sw_version DUT_SW_VERSION]
                     [--dut_model_num DUT_MODEL_NUM]
                     [--test_priority TEST_PRIORITY] [--test_id TEST_ID]
                     [--help_summary]

lf_kpi_csv.py
-----------

Summary :
---------
lf_kpi_csv.py library :

    Date: date of run
    test-rig : testbed that the tests are run on for example ct_us_001
    test-tag : test specific information to differenciate the test,  LANforge radios used, security modes (wpa2 , open)
    dut-hw-version : hardware version of the device under test
    dut-sw-version : software version of the device under test
    dut-model-num : model number / name of the device under test
    dut-serial-num : serial number / serial number of the device under test
    test-priority : test-priority is arbitrary number, choosing under 95 means it goes down at bottom of blog report, and higher p
    test-id : script or test name ,  AP Auto, wifi capacity, data plane, dfs
    short-description : short description of the test
    pass/fail : set blank for performance tests
    numeric-score : this is the value for the y-axis (x-axis is a timestamp),  numeric value of what was measured
    test-details : what was measured in the numeric-score,  e.g. bits per second, bytes per second, upload speed, minimum cx time
    Units : units used for the numeric-scort
    Graph-Group - Items graphed together used by dashboard, For the lf_qa.py dashboard

Example :

    This module is included to assist in filling out the kpi.csv correctly.
    The kpi.csv is used for graphing data over multiple test runs.
    The Unit test is used for helping to become familiar with the library

---------


optional arguments:
  -h, --help            show this help message and exit
```

```
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path, primary use when running test in test suite
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differenciate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --test_id TEST_ID     test-id for kpi.csv,  script or test name
  --help_summary        Show summary of what this script does

           lf_kpi_csv.py : unit test in lf_kpi_csv.py for exersiging lf_kpi_csv.py library
```

py-scripts/lf_logger_config.py

```
usage: lf_logger_config.py [-h] [--help_summary]

This program is a helper  class for setting up python logger
This helper is used by most of the LANforge python scripts.

NAME: lf_logger_config.py

PURPOSE:

This program is a helper  class for setting up python logger

EXAMPLE:
    At top of all files place
    import logging
    logger = logging.getLogger(__name__)

    At top of file that contains the main program. lf_logger_config.py library has
    methods to configure the "root" logger (not linux root).
    DONOT include in base classes

    lf_logger_config = importlib.import_module("py-scripts.lf_logger_config")

    In main program, this will configure the logger with default configuration for all modules
    logger_config = lf_logger_config.lf_logger_config()

    To load a JSON configuration, shown hardcoded as example, uses args to pass in name
    logger_config.lf_logger_config_json = "lf_logger_config.json"
    logger_config.load_lf_logger_config()

    At top of all other files place
    import logging
    logger = logging.getLogger(__name__)

    Then use logger.debug, logger.info, logger.warning, logger.error, logger.critical

    Note: for json the name needs to be the import name py-json.LANforge.LFRequest

    Note: If you attach a handler to a logger and one or more of its ancestors,
        it may emit the same record multiple times. In general,
        you should not need to attach a handler to more than one logger
        - if you just attach it to the appropriate logger which is highest in the logger hierarchy,
        then it will see all events logged by all descendant loggers, provided that their propagate
        setting is left set to True. A common scenario is to attach handlers only to the root logger,
        and to let propagation take care of the rest.

    Additional information:
        https://candelatech.atlassian.net/wiki/spaces/~635833922/pages/982614017/Logging+Lanforge+scripts

LICENSE:
    Free to distribute and modify. LANforge systems must be licensed.
    Copyright 2021 Candela Technologies Inc

INCLUDE_IN_README

optional arguments:
  -h, --help      show this help message and exit
  --help_summary  Show summary of what this script does
```

py-scripts/lf_macvlan.py

```
usage: ./lf_macvlan.py [-h] [--help_summary] [--host HOST]
                       [--parent_port PARENT_PORT] [--new_macvlan]
                       [--mac_pattern MAC_PATTERN] [--qty QTY] [--ip IP]
                       [--state STATE] [--set_state] [--port PORT [PORT ...]]
                       [--rm_macvlan] [--set_ip] [--list] [--debug]
                       [--log_level LOG_LEVEL]

tests creating raw command

optional arguments:
  -h, --help            show this help message and exit
  --help_summary        print out help summary
  --host HOST, --mgr HOST
                        specify the GUI to connect to, assumes port 8080
  --parent_port PARENT_PORT
                        parent port to base macvlans from
  --new_macvlan         create a new macvlan on a parent port
  --mac_pattern MAC_PATTERN
                        MAC address pattern, such as xx:xx:xx:*:*:xx where xx = keep parent, * = random
  --qty QTY             number of macvlans to create
  --ip IP               specify the first IP address with 'ip=<CIDR>,gw=<gateway>' or 'DHCP'
  --state STATE         specify if the port is admin 'up' or admin 'down'
  --set_state           Do not create a macvlan but change the state. Specify if the port is admin 'up' or admin 'down'
  --port PORT [PORT ...]
                        specify the EID of the port to change or remove (--port 1.1.eth2#1 --port 1.1.eth2#0)
  --rm_macvlan, --rm, --del, --remove
                        remove macvlans using --port <EID> arguments
```

```
  --set_ip              Just set IP for a port with --port <EID>,DHCP or --port <EID>,ip=<CIDR>,gw=<IP>
  --list                prints a list of ports, or child ports from --parent_port <EID>
  --debug               turn on debug output
  --log_level LOG_LEVEL
                        specify logging level
```

py-scripts/lf_mesh_test.py

```
usage: lf_mesh_test.py [-h] [-m MGR] [-o PORT] [--lf_user LF_USER]
                       [--lf_password LF_PASSWORD] [-i INSTANCE_NAME]
                       [-c CONFIG_NAME] [-r] [--load_old_cfg]
                       [--enable ENABLE] [--disable DISABLE] [--set SET SET]
                       [--raw_line RAW_LINE] [--raw_lines_file RAW_LINES_FILE]
                       [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                       [-u UPSTREAM] [--download_speed DOWNLOAD_SPEED]
                       [--upload_speed UPLOAD_SPEED] [--duration DURATION]
                       [--graph_groups GRAPH_GROUPS] [--report_dir REPORT_DIR]
                       [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                       [--help_summary]

    Open this file in an editor and read the top notes for more details.

    Example:
    ./lf_mesh_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge \
      --instance_name mesh-instance --config_name test_con --upstream 1.1.eth1 \
      --raw_line 'selected_dut2: RootAP wactest 08:36:c9:19:47:40 (1)' \
      --raw_line 'selected_dut5: RootAP wactest 08:36:c9:19:47:50 (2)' \
      --duration 15s \
      --download_speed 85% --upload_speed 56Kbps \
      --raw_line 'velocity: 100' \
      --raw_lines_file cv_examples/example_cfgs/mesh-ferndale-cfg.txt \
      --test_rig Ferndale-Mesh-01 --pull_report

      NOTE:  There is quite a lot of config needed, see cv_examples/example_cfgs/mesh-ferndale-cfg.txt
         Suggestion is to configure the test through the GUI, make sure it works, then view
         the config and paste it into your own cfg.txt file.



optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the capacity test?  Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).  See example raw text config for possible options.  M
  --disable DISABLE     Specify options to disable (set value to 0).  See example raw text config for possible options.  May be sp
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for instance:  testbed-01
  -u UPSTREAM, --upstream UPSTREAM
                        Upstream port for wifi capacity test ex. 1.1.eth2
  --download_speed DOWNLOAD_SPEED
                        Specify requested download speed.  Percentage of theoretical is also supported.  Default: 85%
  --upload_speed UPLOAD_SPEED
                        Specify requested upload speed.  Percentage of theoretical is also supported.  Default: 0
  --duration DURATION   Specify duration of each traffic run
  --graph_groups GRAPH_GROUPS
                        File to save graph_groups to
  --report_dir REPORT_DIR
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir <where to pull reports to>  default '' put where dataplane script run from
  --help_summary        Show summary of what this script does
```

py-scripts/lf_mixed_traffic.py

```
usage: lf_mixed_traffic.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [--real]
                           [--virtual] [--tests TESTS [TESTS ...]]
                           [--parallel] [--upstream_port UPSTREAM_PORT]
                           [--server_ip SERVER_IP] [--lf_username LF_USERNAME]
                           [--lf_password LF_PASSWORD]
                           [--twog_radio TWOG_RADIO]
                           [--fiveg_radio FIVEG_RADIO]
                           [--sixg_radio SIXG_RADIO]
                           [--twog_security TWOG_SECURITY]
                           [--twog_ssid TWOG_SSID] [--twog_passwd TWOG_PASSWD]
                           [--twog_mode TWOG_MODE]
                           [--twog_num_stations TWOG_NUM_STATIONS]
                           [--twog_start_id TWOG_START_ID]
                           [--fiveg_security FIVEG_SECURITY]
                           [--fiveg_ssid FIVEG_SSID]
                           [--fiveg_passwd FIVEG_PASSWD]
                           [--fiveg_mode FIVEG_MODE]
                           [--fiveg_num_stations FIVEG_NUM_STATIONS]
                           [--fiveg_start_id FIVEG_START_ID]
                           [--sixg_security SIXG_SECURITY]
                           [--sixg_ssid SIXG_SSID] [--sixg_passwd SIXG_PASSWD]
                           [--sixg_mode SIXG_MODE]
                           [--sixg_num_stations SIXG_NUM_STATIONS]
                           [--sixg_start_id SIXG_START_ID]
                           [--dut_model DUT_MODEL]
                           [--dut_firmware DUT_FIRMWARE]
                           [--mixed_traffic_loop MIXED_TRAFFIC_LOOP]
                           [--target TARGET] [--ping_interval PING_INTERVAL]
                           [--traffic_type TRAFFIC_TYPE] [--qos_serial]
                           [--test_duration TEST_DURATION]
                           [--ping_test_duration PING_TEST_DURATION]
```

```
                          [--qos_test_duration QOS_TEST_DURATION]
                          [--ftp_test_duration FTP_TEST_DURATION]
                          [--http_test_duration HTTP_TEST_DURATION]
                          [--multicast_test_duration MULTICAST_TEST_DURATION]
                          [--tos TOS] [--side_a_min SIDE_A_MIN]
                          [--side_a_max SIDE_A_MAX] [--side_b_min SIDE_B_MIN]
                          [--side_b_max SIDE_B_MAX] [--band BAND [BAND ...]]
                          [--use_default_config]
                          [--direction DIRECTION [DIRECTION ...]]
                          [--ftp_file_sizes FTP_FILE_SIZES [FTP_FILE_SIZES ...]]
                          [--target_per_ten TARGET_PER_TEN]
                          [--http_file_size HTTP_FILE_SIZE]
                          [--mc_traffic_type MC_TRAFFIC_TYPE]
                          [--mc_tos MC_TOS] [--side_a_min_bps SIDE_A_MIN_BPS]
                          [--side_a_min_pdu SIDE_A_MIN_PDU]
                          [--side_b_min_bps SIDE_B_MIN_BPS]
                          [--side_b_min_pdu SIDE_B_MIN_PDU]
                          [--polling_interval POLLING_INTERVAL]
                          [--pre_cleanup] [--all_bands] [--dowebgui]
                          [--device_list DEVICE_LIST]
                          [--result_dir RESULT_DIR] [--test_name TEST_NAME]
                          [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                          [--help_summary]


NAME: lf_mixed_traffic.py

PURPOSE:
        Mixed traffic test is designed to measure the access point performance and stability by running multiple traffic
        on both virtual & real clients like Android, Linux, Windows, and IOS connected to the access point.
        This test allows the user to choose multiple types of traffic like client ping test, qos test, ftp test, http test,
        multicast test and run tests both serially and simultaneously.

EXAMPLE:

        # CLI for Mixed Traffic Test: Run with Real Clients without connecting to a specified SSID

            python3 lf_mixed_traffic.py --mgr 192.168.212.100 --tests 1 2 3 4 5 --target 10.0.0.10 --ping_interval 5 --upstream_po
            --side_b_min_bps 3000000 --side_a_min 1000000 --side_b_min 1000000 --traffic_type lf_tcp --tos "VI"
            --ftp_file_sizes 10MB --http_file_size 5MB --direction Download --mc_tos "BE" --real --qos_serial --mixed_traffic_loop
            --ping_test_duration 1m --qos_test_duration 30s --ftp_test_duration 30s --http_test_duration 30s --multicast_test_dura
            --pre_cleanup

        # CLI for Mixed Traffic Test: Run with Real Clients with parallel execution of tests

            python3 lf_mixed_traffic.py --mgr 192.168.212.100
            --twog_ssid NETGEAR-2G --twog_passwd Password@123 --twog_security wpa2 --twog_radio wiphy0 --twog_num_stations 5
            --fiveg_ssid NETGEAR-5G --fiveg_passwd Password@123 --fiveg_security wpa2 --fiveg_radio wiphy1 --fiveg_num_stations 5
            --band 2.4G,5G --tests 1 2 3 4 5 --target 10.0.0.10 --ping_interval 5 --upstream_port 1.1.eth1
            --side_b_min_bps 3000000 --side_a_min 1000000 --side_b_min 1000000 --traffic_type lf_tcp --tos "VI"
            --ftp_file_sizes 10MB --http_file_size 5MB --direction Download --mc_tos "BE" --real --mixed_traffic_loop 1
            --ping_test_duration 1m --qos_test_duration 30s --ftp_test_duration 30s --http_test_duration 30s --multicast_test_dura
            --pre_cleanup --parallel

        # CLI for Mixed Traffic Test: Run with Real Clients on 2.4GHz & 5GHz Bands, Single Iteration per Band.

            python3 lf_mixed_traffic.py --mgr 192.168.212.100
            --twog_ssid NETGEAR-2G --twog_passwd Password@123 --twog_security wpa2 --twog_radio wiphy0 --twog_num_stations 5
            --fiveg_ssid NETGEAR-5G --fiveg_passwd Password@123 --fiveg_security wpa2 --fiveg_radio wiphy1 --fiveg_num_stations 5
            --band 2.4G,5G --tests 1 2 3 4 5 --target 10.0.0.10 --ping_interval 5 --upstream_port 1.1.eth1
            --side_b_min_bps 3000000 --side_a_min 1000000 --side_b_min 1000000 --traffic_type lf_tcp --tos "VI"
            --ftp_file_sizes 10MB --http_file_size 5MB --direction Download --mc_tos "BE" --real --qos_serial --mixed_traffic_loop
            --ping_test_duration 1m --qos_test_duration 30s --ftp_test_duration 30s --http_test_duration 30s --multicast_test_dura
            --pre_cleanup

        # CLI for Mixed Traffic Test: Run on 2.4GHz & 5GHz Bands Simultaneously with Real Clients in a Single Iteration.

            python3 lf_mixed_traffic.py --mgr 192.168.212.100
            --twog_ssid NETGEAR-2G --twog_passwd Password@123 --twog_security wpa2 --twog_radio wiphy0 --twog_num_stations 5
            --fiveg_ssid NETGEAR-5G --fiveg_passwd Password@123 --fiveg_security wpa2 --fiveg_radio wiphy1 --fiveg_num_stations 5
            --band 2.4G,5G --tests 1 2 3 4 5 --target 10.0.0.10 --ping_interval 5 --upstream_port 1.1.eth1
            --side_b_min_bps 3000000 --side_a_min 1000000 --side_b_min 1000000 --traffic_type lf_tcp --tos "VI"
            --ftp_file_sizes 10MB --http_file_size 5MB --direction Download --mc_tos "BE" --real --qos_serial --mixed_traffic_loop
            --ping_test_duration 1m --qos_test_duration 30s --ftp_test_duration 30s --http_test_duration 30s --multicast_test_dura
            --all_bands --pre_cleanup

        # CLI for Mixed Traffic Test: Run with Virtual Clients on 2.4GHz & 5GHz Bands, Single Iteration per Band.

            python3 lf_mixed_traffic.py --mgr 192.168.212.100
            --twog_ssid NETGEAR-2G --twog_passwd Password@123 --twog_security wpa2 --twog_radio wiphy0 --twog_num_stations 5
            --fiveg_ssid NETGEAR-5G --fiveg_passwd Password@123 --fiveg_security wpa2 --fiveg_radio wiphy1 --fiveg_num_stations 5
            --band 2.4G,5G --tests 1 2 3 4 5 --target 10.0.0.10 --ping_interval 5 --upstream_port 1.1.eth1
            --side_b_min_bps 3000000 --side_a_min 1000000 --side_b_min 1000000 --traffic_type lf_tcp --tos "VI"
            --ftp_file_sizes 10MB --http_file_size 5MB --direction Download --mc_tos "BE" --virtual --qos_serial --mixed_traffic_l
            --ping_test_duration 1m --qos_test_duration 30s --ftp_test_duration 30s --http_test_duration 30s --multicast_test_dura
            --pre_cleanup

        # CLI for Mixed Traffic Test: Run on 2.4GHz & 5GHz Bands Simultaneously with Virtual Clients in a Single Iteration.

            python3 lf_mixed_traffic.py --mgr 192.168.212.100
            --twog_ssid NETGEAR-2G --twog_passwd Password@123 --twog_security wpa2 --twog_radio wiphy0 --twog_num_stations 5
            --fiveg_ssid NETGEAR-5G --fiveg_passwd Password@123 --fiveg_security wpa2 --fiveg_radio wiphy1 --fiveg_num_stations 5
            --band 2.4G,5G --tests 1 2 3 4 5 --target 10.0.0.10 --ping_interval 5 --upstream_port 1.1.eth1
            --side_b_min_bps 3000000 --side_a_min 1000000 --side_b_min 1000000 --traffic_type lf_tcp --tos "VI"
            --ftp_file_sizes 10MB --http_file_size 5MB --direction Download --mc_tos "BE" --virtual --qos_serial --mixed_traffic_l
            --ping_test_duration 1m --qos_test_duration 30s --ftp_test_duration 30s --http_test_duration 30s --multicast_test_dura
            --all_bands --pre_cleanup

SCRIPT_CLASSIFICATION:  Multiples Tests, Creation, Report Generation (Both individual & Overall)

SCRIPT_CATEGORIES:  Performance, Functional

NOTES:
        The primary goal of the script is to execute a series of tests and group their individual test reports into a unified repo

STATUS: Functional
```

```
VERIFIED_ON: 02-May-2024
            GUI Version:  5.4.8
            Build Date :  Sun Apr 21 01:42:42 PM PDT 2024
            Kernel Version: 6.2.16+

LICENSE:
         Free to distribute and modify. LANforge systems must be licensed.
         Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False

optional arguments:
  -h, --help            show this help message and exit
  --target_per_ten TARGET_PER_TEN
                        number of request per 10 minutes
  --http_file_size HTTP_FILE_SIZE
                        specify the size of file you want to download
  --mc_traffic_type MC_TRAFFIC_TYPE
                        multicast tos
  --mc_tos MC_TOS       multicast tos
  --side_a_min_bps SIDE_A_MIN_BPS
                        --side_a_min_bps, requested downstream min tx rate, comma separated list for multiple iterations.  Default
  --side_a_min_pdu SIDE_A_MIN_PDU
                        --side_a_min_pdu, downstream pdu size, comma separated list for multiple iterations.  Default MTU
  --side_b_min_bps SIDE_B_MIN_BPS
                        --side_b_min_bps, requested upstream min tx rate, comma separated list for multiple iterations.  Default 2
  --side_b_min_pdu SIDE_B_MIN_PDU
                        --side_b_min_pdu, upstream pdu size, comma separated list for multiple iterations. Default MTU
  --polling_interval POLLING_INTERVAL
                        --polling_interval <seconds>
  --pre_cleanup         Use this if you want to clean Generic, Layer-3, L3 Endps & Layer 4-7 tabs data
  --all_bands           to run the tests with respective bands
  --dowebgui            to run the tests with webgui
  --device_list DEVICE_LIST
                        device list recieved from webgui tos
  --result_dir RESULT_DIR
                        result_dir for real time data for webui
  --test_name TEST_NAME
                        test name for webui
  --help_summary        Show summary of what this script does

Required arguments to run lf_mixed_traffic.py:
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --real                Enable this flag to run the tests on real clients
  --virtual             Enable this flag to run the tests on virtual clients
  --tests TESTS [TESTS ...]
                        To enable Tests
                        1 --> Ping Test
                        2 --> Qos Test
                        3 --> FTP Test
                        4 --> HTTP Test
                        5 --> Multicast Test
                        eg: --tests 1 2 3 4 5
  --target TARGET       Target URL for ping test
  --ping_interval PING_INTERVAL
                        Interval (in seconds) between the echo requests
  --direction DIRECTION [DIRECTION ...]
                        Enter the traffic direction. Example : "Download","Upload"
  --ftp_file_sizes FTP_FILE_SIZES [FTP_FILE_SIZES ...]
                        File Size Example : "1000MB"

Optional arguments to run lf_mixed_traffic.py:
  --parallel            Use --parallel to run the all the traffics (ftp,http,qos,ping,multicast..) simultaeneously
  --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --server_ip SERVER_IP
                        specify the server ip to login to the app
  --lf_username LF_USERNAME
                        Specify the lanforge ssh user name
  --lf_password LF_PASSWORD
                        Specify the lanforge ssh password
  --twog_radio TWOG_RADIO
                        specify radio for 2.4G clients [default = wiphy1]
  --fiveg_radio FIVEG_RADIO
                        specify radio for 5G client [default = wiphy0]
  --sixg_radio SIXG_RADIO
                        specify radio for 6G clients [default = wiphy2]
  --twog_security TWOG_SECURITY
                        WiFi Security protocol: {open|wep|wpa2|wpa3} for 2.4G clients
  --twog_ssid TWOG_SSID
                        WiFi SSID for script object to associate for 2.4G clients
  --twog_passwd TWOG_PASSWD
                        WiFi passphrase/password/key for 2.4G clients
  --twog_mode TWOG_MODE
                        Mode for your twog station (as a number)
  --twog_num_stations TWOG_NUM_STATIONS
                        number of virtual stations for twog band
  --twog_start_id TWOG_START_ID
                        Specify the station starting id for twog stations.
                         e.g: --twog_start_id <value> default 2000
  --fiveg_security FIVEG_SECURITY
                        WiFi Security protocol: {open|wep|wpa2|wpa3} for 5G clients
  --fiveg_ssid FIVEG_SSID
                        WiFi SSID for script object to associate for 5G clients
  --fiveg_passwd FIVEG_PASSWD
                        WiFi passphrase/password/key for 5G clients
  --fiveg_mode FIVEG_MODE
                        Mode for your fiveg station (as a number)
  --fiveg_num_stations FIVEG_NUM_STATIONS
                        number of virtual stations for fiveg band
  --fiveg_start_id FIVEG_START_ID
                        Specify the station starting id for fiveg stations.
                         e.g: --fiveg_start_id <value> default 5000
  --sixg_security SIXG_SECURITY
                        WiFi Security protocol: {open|wep|wpa2|wpa3} for 6G clients
  --sixg_ssid SIXG_SSID
```

```
                        WiFi SSID for script object to associate for 6G clients
  --sixg_passwd SIXG_PASSWD
                        WiFi passphrase/password/key for 6G clients
  --sixg_mode SIXG_MODE
                        Mode for your sixg station (as a number)
  --sixg_num_stations SIXG_NUM_STATIONS
                        number of virtual stations for sixg band
  --sixg_start_id SIXG_START_ID
                        Specify the station starting id for sixg stations.
                         e.g: --sixg_start_id <value> default 6000
  --dut_model DUT_MODEL
                        Specify the Dut Name. eg: --dut_model EAP101
  --dut_firmware DUT_FIRMWARE
                        Specify the dut firmware. eg: --dut_firmware V1.0.0.10
  --mixed_traffic_loop MIXED_TRAFFIC_LOOP
                        Specify the number of times mixed traffic test should run
  --traffic_type TRAFFIC_TYPE
                        Select the Traffic Type [lf_udp, lf_tcp]
  --qos_serial          Use --qos_serial to run the qos test one after another.
  --test_duration TEST_DURATION
                        --test_duration sets the duration for all the tests
  --ping_test_duration PING_TEST_DURATION
                        --test_duration sets the duration for ping tests
  --qos_test_duration QOS_TEST_DURATION
                        --test_duration sets the duration for qos tests
  --ftp_test_duration FTP_TEST_DURATION
                        --test_duration sets the duration for ftp tests
  --http_test_duration HTTP_TEST_DURATION
                        --test_duration sets the duration for http tests
  --multicast_test_duration MULTICAST_TEST_DURATION
                        --test_duration sets the duration for multicast tests
  --tos TOS             Enter the tos. Example1 : "BK,BE,VI,VO" , Example2 : "BK,VO", Example3 : "VI"
  --side_a_min SIDE_A_MIN
                        Endpoint-a Min Cx Rate
  --side_a_max SIDE_A_MAX
                        Endpoint-a Max CX Rate
  --side_b_min SIDE_B_MIN
                        Endpoint-b Min CX Rate
  --side_b_max SIDE_B_MAX
                        Endpoint-b Max CX Rate
  --band BAND [BAND ...]
                        select bands for virtual clients Example : "5G","2.4G"
  --use_default_config  specify this flag if wanted to proceed with existing Wi-Fi configuration of the devices
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger

               The Mixed Traffic Test is Used for running the different test in single test
```

py-scripts/lf_modify_radio.py

```
usage: ./lf_modify_radio.py [-h] [--host MGR] [--mgr_port MGR_PORT]
                            [--lf_user LF_USER] [--lf_passwd LF_PASSWD]
                            [--radio RADIO] [--antenna ANTENNA]
                            [--channel CHANNEL] [--txpower TXPOWER]
                            [--country {US,AL,DZ,AR,BD,AM,AU,AT,AZ,BH,BB,BY,BE,BZ,BO,BA,BR,BN,BG,CA,CL,CN,CO,CR,HR,CY,CZ,DK,DO,EC,
                            [--log_level LOG_LEVEL]
                            [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                            [--debug] [--enable_dhcp ENABLE_DHCP]
                            [--static STATIC] [--static_ip STATIC_IP]
                            [--ip_mask IP_MASK] [--gateway_ip GATEWAY_IP]
                            [--help_summary]

       modifies radio configuration , antenna, radio, channel or txpower
           lf_modify_radio.py --mgr 192.168.0.104 --radio 1.1.wiphy6 --txpower 17 --debug


optional arguments:
  -h, --help            show this help message and exit
  --host MGR, --mgr MGR
                        specify the GUI to connect to
  --mgr_port MGR_PORT   specify the GUI to connect to, default 8080
  --lf_user LF_USER     lanforge user name, default : lanforge
  --lf_passwd LF_PASSWD
                        lanforge password, default : lanforge
  --radio RADIO         name of the radio to modify: e.g. 1.1.wiphy0
  --antenna ANTENNA     number of spatial streams: 0 Diversity (All), 1 Fixed-A (1x1), 4 AB (2x2), 7 ABC (3x3), 8 ABCD (4x4), 9 (8
  --channel CHANNEL     channel of the radio: e.g. 6 (2.4G) or 36 (5G) default: AUTO
  --txpower TXPOWER     radio tx power default: AUTO system defaults
  --country {US,AL,DZ,AR,BD,AM,AU,AT,AZ,BH,BB,BY,BE,BZ,BO,BA,BR,BN,BG,CA,CL,CN,CO,CR,HR,CY,CZ,DK,DO,EC,EG,SV,EE,FI,FR,GE,DE,GR,GT,
                        Set the country code for the lanforge resource. This needs to set all radios on the resource to the same c
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --debug               Legacy debug flag, turnn on legacy debug
  --enable_dhcp ENABLE_DHCP
                        set to True if wanted to enbale DHCP-IPv4 on eth interface
  --static STATIC       True if client will be created with static ip
  --static_ip STATIC_IP
                        if static option is True provide static ip to client
  --ip_mask IP_MASK     if static is true provide ip mask to client
  --gateway_ip GATEWAY_IP
                        if static is true provide gateway ip
  --help_summary        Show summary of what this script does
```

py-scripts/lf_multipsk.py

```
usage: lf_multipsk.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                      [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                      [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                      [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                      [--proxy [PROXY]]
                      [--debugging DEBUGGING [DEBUGGING ...]]
                      [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
                      [--radio RADIO] [--security SECURITY] [--ssid SSID]
                      [--passwd PASSWD] [--n_vlan N_VLAN] [--mode MODE]
```

```
NAME: lf_multipsk.py

PURPOSE:
        to test the multipsk feature in access point. Multipsk feature states connecting clients using same ssid but different pas
        here we will create two or 3 passwords with different vlan id on single ssid and try to connect client with different pass

        NOTE:  This script has a lot of hard-coded values, and will only work in a very particular setup
        like what is used in TIP labs with APs set up for VLANs and LANforge set up to serve DHCP address
        on un-tagged ports as well as tagged 1q vlans.  This script may be improved in the future to be more flexible.

        NOTE:  This script does not create the .1q vlan interfaces, it assumes they already exist before this script
        runs.

DESCRIPTION:
            The script will follow basic functionality as:-
            1- create station on input parameters provided
            2- the input parameters consist of dictionary of passwords,upstream,mac address, number of clients and radio as input
            3- will create layer3 cx for tcp and udp
            3- verify layer3 cx
            4- verify the ip for each station is comming from respective vlan id or not.
example :-
         python3 lf_multipsk.py --mgr localhost  --mgr_port 8802  --ssid "MDU Wi-Fi"  --security wpa2

INCLUDE_IN_README
    -Nikita Yadav
    Copyright 2021 Candela Technologies Inc
    License: Free to distribute and modify. LANforge systems must be licensed.

optional arguments:
  -h, --help            show this help message and exit
  --n_vlan N_VLAN       type number of vlan using in test eg 1 or 2
  --mode MODE           Mode for lf_multipsk

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key
```

py-scripts/lf_pcap.py

```
usage: lf_pcap.py [-h] [--pcap_file PCAP_FILE] [--apply_filter APPLY_FILTER]
                  [--help_summary]

    """
-----------------------
NAME: lf_pcap.py

PURPOSE:
Common Library for reading pcap files and check packet information for specific filters

SETUP:
This script requires pyshark to be installed before,you can install it by running "pip install pyshark"

EXAMPLE:
see: /py-scritps/lf_pcap.py
----------------------

optional arguments:
  -h, --help            show this help message and exit
  --pcap_file PCAP_FILE, -p PCAP_FILE
                        provide the pcap file path
  --apply_filter APPLY_FILTER, -f APPLY_FILTER
                        apply the filter you want to
  --help_summary        Show summary of what this script does

Common Library for reading pcap files and check packet information for specific filters
```

py-scripts/lf_ping_sweep.py

```
usage: lf_ping_sweep.py [-h] [-e ETHERNET_INTERFACE] [-ip TARGET_IP]
                        [-dur DURATION] [-csv CSV_NAME]
                        [-data_len DATA_LENGTHS [DATA_LENGTHS ...]]
                        [-admin_password ADMIN_PASS]
                        [-reporting_down_time_percentage REPORTING_DOWN_TIME_PERCENTAGE]
```

```
                        [--help_summary]

----------------------------------------------------------------------------------------------------
Objective:
    This Ping Sweep Test script is designed to discover devices within the network
    connectivity by measuring latency. It also detects issues like client unavailability time,
    and average latency, ensuring effective device communication and identifying
    connectivity problems
    Note:
        The script uses Nmap tool.
        If Namp not present in the system , then it can be installed using the command: sudo apt-get install nmap
----------------------------------------------------------------------------------------------------
CLI Example1:
    python3 lf_ping_sweep.py -e eth1 -ip 192.168.1.1/24  --duration 10h --csv_name ping_sweep_1 --reporting_down_time_percentage 3

CLI Example2 - with data packets:
    python3 lf_ping_sweep.py -e eth1 -ip 192.168.1.1/24 --data_lengths 32 64 128 500  --duration 10m --csv_name ping_sweep_1 --rep

CLI Example3 - with different root password:
    python3 lf_ping_sweep.py -e eth1 -ip 192.168.1.1/24 --data_lengths 32 64 1024  --duration 1h --csv_name ping_sweep_1  --admin_

Note: --reporting_down_time_percentage is used to calculate the ips which were down more than given percentage of its duration (de

----------------------------------------------------------------------------------------------------

optional arguments:
  -h, --help            show this help message and exit
  -e ETHERNET_INTERFACE, --ethernet_interface ETHERNET_INTERFACE
                        ethernet interface
  -ip TARGET_IP, --target_ip TARGET_IP
                        mention the target ip or network subnet
  -dur DURATION, --duration DURATION
                        provide the scanning duration in either hours or minutes or seconds,ex: 2h or 30m or 50s
  -csv CSV_NAME, --csv_name CSV_NAME
                        provide the file name  in which the csv should be created
  -data_len DATA_LENGTHS [DATA_LENGTHS ...], --data_lengths DATA_LENGTHS [DATA_LENGTHS ...]
                        List of data lengths
  -admin_password ADMIN_PASS, --admin_pass ADMIN_PASS
                        provide the admin/root password
  -reporting_down_time_percentage REPORTING_DOWN_TIME_PERCENTAGE, --reporting_down_time_percentage REPORTING_DOWN_TIME_PERCENTAGE
                        provide the percentage for calculation of ips which were down more than the percentage of duration, defaul
  --help_summary        Show summary of what this script does
```

py-scripts/lf_report.py

```
usage: lf_report.py [-h] [--lfmgr LFMGR] [--help_summary]

Reporting library Unit Test

optional arguments:
  -h, --help      show this help message and exit
  --lfmgr LFMGR   sample argument: where LANforge GUI is running
  --help_summary  Show summary of what this script does
```

py-scripts/lf_report_test.py

```
usage: lf_report_test.py [-h] [--mgr LFMGR] [--help_summary]

-----------------
NAME: lf_report_test.py

PURPOSE:
Common file for testing lf_report and lf_graph Library generates html and pdf output

SETUP:
/lanforge/html-reports directory needs to be present or output generated in local file

EXAMPLE:
./lf_report_test.py : currently script does not accept input

COPYWRITE
    Copyright 2021 Candela Technologies Inc
    License: Free to distribute and modify. LANforge systems must be licensed.

INCLUDE_IN_README

optional arguments:
  -h, --help            show this help message and exit
  --mgr LFMGR, --lfmgr LFMGR
                        sample argument: where LANforge GUI is running
  --help_summary        provides help_summary to understand the script
```

py-scripts/lf_rf_char.py

```
usage: lf_rf_char.py [-h] [--mgr LF_MGR] [--mgr_port LF_PORT]
                     [--lf_user LF_USER] [--lf_passwd LF_PASSWD]
                     [--vap_port VAP_PORT] [--vap_radio VAP_RADIO]
                     [--vap_bw {40,20,160,80}] [--vap_channel VAP_CHANNEL]
                     [--vap_antenna {all,8x8,3x3,2x2,1x1,4x4}]
                     [--vap_mode VAP_MODE] [--vap_txpower VAP_TXPOWER]
                     [--reset_vap] [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                     [--final_report_dir FINAL_REPORT_DIR] [--no_pdf]
                     [--no_html] [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                     [--dut_hw_version DUT_HW_VERSION]
                     [--dut_sw_version DUT_SW_VERSION]
                     [--dut_model_num DUT_MODEL_NUM]
                     [--dut_serial_num DUT_SERIAL_NUM]
                     [--test_priority TEST_PRIORITY] [--test_id TEST_ID]
                     [--csv_outfile CSV_OUTFILE] [--log_level LOG_LEVEL]
                     [--lf_logger_config_json LF_LOGGER_CONFIG_JSON] [--debug]
                     [--desc DESC] [--polynomial] [--interpolate]
                     [--duration DURATION]
                     [--polling_interval POLLING_INTERVAL] [--frame FRAME]
                     [--frame_interval FRAME_INTERVAL]
                     [--timeout_sec TIMEOUT_SEC] [--dhcp_poll_ms DHCP_POLL_MS]
                     [--dhcp_lookup_attempts DHCP_LOOKUP_ATTEMPTS]
                     [--help_summary]
```

```
Summary :
---------
Gather Tx and Rx RF Characteristic for a specific duration and polling interval

Example :
---------
./lf_rf_char.py     --lf_mgr            192.168.0.104     --vap_port          1.1.vap3     --vap_radio         1.1.wiphy3     --va

for individual command telnet <lf_mgr> 4001 ,  then can execute cli commands


optional arguments:
  -h, --help            show this help message and exit
  --mgr LF_MGR, --lf_mgr LF_MGR
                        address of the LANforge GUI machine (localhost is default)
  --mgr_port LF_PORT, --lf_port LF_PORT
                        IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     user: lanforge
  --lf_passwd LF_PASSWD
                        passwd: lanforge
  --vap_port VAP_PORT   port: 1.1.vap3  provide full eid (endpoint id)
  --vap_radio VAP_RADIO
                        --vap_radio wiphy0
  --vap_bw {40,20,160,80}
                        Specify the bandwidth for the vAP. Not all bandwidth settings are available for all radios.
  --vap_channel VAP_CHANNEL
                        Specify the channel of the radio e.g. 6 (2.4G), 36 (5G), 1e (6G) Please append 'e' for all 6Ghz channels.
  --vap_antenna {all,8x8,3x3,2x2,1x1,4x4}
                        Spatial stream configuration of vAP. Support for each configuration depends on the radio used.
  --vap_mode VAP_MODE   WiFi modes defined by www.candelatech.com/lfcli_ug.php#add_vap Includes 802.11a, a, b, g, abg, abgn, bgn,
  --vap_txpower VAP_TXPOWER
                        Transmit power used by vAP radio measured in dBm, e.g. '25dBm'Accepted values are 'DEFAULT' or an integer
  --reset_vap           Specify this if DHCP leases do not disappear from the vAP. Default behavior is to not reset the vAP
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path, primary use when running test in test suite. The lowest ac
  --final_report_dir FINAL_REPORT_DIR
                        moves the default report directory (mv <time-date>_rf_characteristics) to new name. If this is not a full
  --no_pdf              specify this to skip PDF generation
  --no_html             specify this to skip generating a HTML report and charts, only csv output would be created. Implies --no_p
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv, test specific information to differenciate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM, --dut_model_no DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM, --dut_serial_no DUT_SERIAL_NUM
                        dut serial num for kpi.csv,  model serial number
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv, test-priority is arbitrary number
  --test_id TEST_ID     test-id for kpi.csv,  script or test name
  --csv_outfile CSV_OUTFILE
                        csv outfile
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --debug               Legacy debug flag
  --desc DESC           --desc <test description> , if not provided will section not printed
  --polynomial          --polynomial store_true , show polynomial lines on retries graph
  --interpolate         --interpolate store_true , show interpolation on retries graph
  --duration DURATION   --duration <seconds>
  --polling_interval POLLING_INTERVAL
                        --polling_interval <h m s ms>
  --frame FRAME         --frame <bytes>  , e.g. --frame 1400
  --frame_interval FRAME_INTERVAL
                        --frame_interval <fractions of second>  , e.g. --frame_interval .01
  --timeout_sec TIMEOUT_SEC
                        number of seconds to allow for starting traffic; if traffic has not started by this time, script will abor
  --dhcp_poll_ms DHCP_POLL_MS
                        wait between checking vap probe results for new DHCP leases (milliseconds)
  --dhcp_lookup_attempts DHCP_LOOKUP_ATTEMPTS
                        number of attempts to check for DHCP lease before aborting script
  --help_summary        Show summary of what this script does

           lf_rf_char.py : RF Characteristics test
```

py-scripts/lf_rfgen_info.py

```
usage: lf_rfgen_info.py [-h] [--mgr MGR] [--port PORT] [--log_level LOG_LEVEL]
                        [--help_summary]

NAME:
lf_rfgen_info.py

PURPOSE:
Read the rfgen_info from lanforge
May be used as a module

EXAMPLE:

./lf_rfgen_info --mgr <ip>

NOTES:

# https://docs.python.org/3.8/library/telnetlib.html



optional arguments:
  -h, --help            show this help message and exit
  --mgr MGR             --mgr <lanforge ip>
  --port PORT           --port <lanforge telnet port>
  --log_level LOG_LEVEL
```

```
                      Set logging level: debug | info | warning | error | critical
  --help_summary      Show summary of what this script does

           lf_rfgen_info.py
```

py-scripts/lf_roam_test.py

```
usage: lf_roam_test.py [-h] [--mgr MGR] [--lanforge_port LANFORGE_PORT]
                       [--lanforge_ssh_port LANFORGE_SSH_PORT]
                       [--ap1_bssid AP1_BSSID] [--ap2_bssid AP2_BSSID]
                       [--twog_radios TWOG_RADIOS]
                       [--fiveg_radios FIVEG_RADIOS]
                       [--sixg_radios SIXG_RADIOS] [--band BAND]
                       [--sniff_radio SNIFF_RADIO] [--num_sta NUM_STA]
                       [--ssid_name SSID_NAME] [--security SECURITY]
                       [--security_key SECURITY_KEY] [--upstream UPSTREAM]
                       [--duration DURATION] [--iteration ITERATION]
                       [--channel CHANNEL] [--option OPTION]
                       [--iteration_based] [--duration_based]
                       [--dut_name DUT_NAME] [--traffic_type TRAFFIC_TYPE]
                       [--identity IDENTITY] [--ttls_pass TTLS_PASS]
                       [--log_file LOG_FILE] [--debug DEBUG]
                       [--soft_roam SOFT_ROAM] [--sta_type STA_TYPE]
                       [--multicast MULTICAST] [--scheme SCHEME] [--dest DEST]
                       [--user USER] [--passwd PASSWD] [--prompt PROMPT]
                       [--series_cc SERIES_CC] [--ap AP]
                       [--ap_ssh_port AP_SSH_PORT] [--band_cc BAND_CC]
                       [--timeout TIMEOUT] [--help_summary]

lf_roam_test.py :
--------------------

Summary :
----------
The primary focus of this script is to enable seamless roaming of clients/stations between two access points (APs).
The test can be conducted with a single or multiple stations, with single or multiple iterations.

The script will create stations/clients with advanced/802.1x and 11r key management. By default, it will create a
single station/client. Once the stations are created, the script will generate CX traffic between the upstream port and
 the stations and run the traffic before roam.

Packet captures will be taken for each station/client in two scenarios:

    (i)  While the station/client is connected to an AP
    (ii) While the station/client roams from one AP to another AP

These packet captures will be used to analyze the performance and stability of the roaming process.

Overall, this script is designed to provide a comprehensive test of the roaming functionality of the APs and the
stability of the network when clients move between APs.

 The following are the criteria for PASS the test:

    1. The BSSID of the station should change after roaming from one AP to another
    2  The station should not experience any disconnections during/after the roaming process.
    3. The duration of the roaming process should be less than 50 ms.

 The following are the criteria for FAIL the test:

    1. The BSSID of the station remains unchanged after roaming from one AP to another.
    2. No roaming occurs, as all stations are connected to the same AP.
    3. The captured packet does not contain a Reassociation Response Frame.
    4. The station experiences disconnection during/after the roaming process.
    5. The duration of the roaming process exceeds 50 ms.

##########################################
# Examples Commands for different scenarios
##########################################

Hard Roam

EXAMPLE: For a single station and a single iteration
    python3 lf__roam_test.py --mgr 192.168.100.221 --ap1_bssid "68:7d:b4:5f:5c:3b" --ap2_bssid "14:16:9d:53:58:cb"
     --fiveg_radios "1.1.wiphy1" --band "fiveg" --sniff_radio "wiphy2" --num_sta 1 --ssid_name "RoamAP5g" --security "wpa2"
      --security_key "something" --duration None --upstream "eth2" --iteration 1 --channel "40" --option "ota"
      --dut_name ["AP1","AP2"] --traffic_type "lf_udp" --log_file False --debug False --iteration_based

EXAMPLE: For a single station and multiple iteration
    python3 lf_roam_test.py --mgr 192.168.100.221 --ap1_bssid "68:7d:b4:5f:5c:3b" --ap2_bssid "14:16:9d:53:58:cb"
     --fiveg_radios "1.1.wiphy1" --band "fiveg" --sniff_radio "wiphy2" --num_sta 1 --ssid_name "RoamAP5g" --security "wpa2"
      --security_key "something" --duration None --upstream "eth2" --iteration 10 --channel "40" --option "ota"
      --dut_name ["AP1","AP2"] --traffic_type "lf_udp" --log_file False --debug False --iteration_based

EXAMPLE: For multiple station and a single iteration
    python3 lf_roam_test.py --mgr 192.168.100.221 --ap1_bssid "68:7d:b4:5f:5c:3b" --ap2_bssid "14:16:9d:53:58:cb"
     --fiveg_radios "1.1.wiphy1" --band "fiveg" --sniff_radio "wiphy2" --num_sta 10 --ssid_name "RoamAP5g" --security "wpa2"
      --security_key "something" --duration None --upstream "eth2" --iteration 1 --channel "40" --option "ota"
      --dut_name ["AP1","AP2"] --traffic_type "lf_udp" --log_file False --debug False --iteration_based

EXAMPLE: For multiple station and multiple iteration
    python3 lf_roam_test.py --mgr 192.168.100.221 --ap1_bssid "68:7d:b4:5f:5c:3b" --ap2_bssid "14:16:9d:53:58:cb"
     --fiveg_radios "1.1.wiphy1" --band "fiveg" --sniff_radio "wiphy2" --num_sta 10 --ssid_name "RoamAP5g" --security "wpa2"
      --security_key "something" --duration None --upstream "eth2" --iteration 10 --channel "40" --option "ota"
      --dut_name ["AP1","AP2"] --traffic_type "lf_udp" --log_file False --debug False --iteration_based

EXAMPLE: For  multiple station and multiple iteration with multicast traffic enable
   python3 lf_roam_test.py --mgr 192.168.100.221 --ap1_bssid "10:f9:20:fd:f3:4b" --ap2_bssid "14:16:9d:53:58:cb"
   --fiveg_radios "1.1.wiphy1" --band "fiveg" --sniff_radio "wiphy2" --num_sta 2 --ssid_name "RoamAP5g" --security "wpa2"
     --security_key "something" --duration None --upstream "eth2" --iteration 1 --channel "36" --option "ota"
      --dut_name ["AP1","AP2"] --traffic_type "lf_udp" --log_file False --debug False --iteration_based --sta_type normal --multic

Soft Roam
EXAMPLE: For a single station and a single iteration
    python3 lf_roam_test.py --mgr 192.168.100.221 --ap1_bssid "68:7d:b4:5f:5c:3b" --ap2_bssid "14:16:9d:53:58:cb"
     --fiveg_radios "1.1.wiphy1" --band "fiveg" --sniff_radio "wiphy2" --num_sta 1 --ssid_name "RoamAP5g" --security "wpa2"
      --security_key "something" --duration None --upstream "eth2" --iteration 1 --channel "40" --option "ota"
```

```
     --dut_name ["AP1","AP2"] --traffic_type "lf_udp" --log_file False --debug False --iteration_based --soft_roam True

EXAMPLE: For a single station and multiple iteration
    python3 lf_roam_test.py --mgr 192.168.100.221 --ap1_bssid "68:7d:b4:5f:5c:3b" --ap2_bssid "14:16:9d:53:58:cb"
     --fiveg_radios "1.1.wiphy1" --band "fiveg" --sniff_radio "wiphy2" --num_sta 1 --ssid_name "RoamAP5g" --security "wpa2"
      --security_key "something" --duration None --upstream "eth2" --iteration 10 --channel "40" --option "ota"
      --dut_name ["AP1","AP2"] --traffic_type "lf_udp" --log_file False --debug False --iteration_based --soft_roam True

EXAMPLE: For multiple station and a single iteration
    python3 lf_roam_test.py --mgr 192.168.100.221 --ap1_bssid "68:7d:b4:5f:5c:3b" --ap2_bssid "14:16:9d:53:58:cb"
     --fiveg_radios "1.1.wiphy1" --band "fiveg" --sniff_radio "wiphy2" --num_sta 10 --ssid_name "RoamAP5g" --security "wpa2"
      --security_key "something" --duration None --upstream "eth2" --iteration 1 --channel "40" --option "ota"
      --dut_name ["AP1","AP2"] --traffic_type "lf_udp" --log_file False --debug False --iteration_based --soft_roam True

EXAMPLE: For multiple station and multiple iteration
    python3 lf_roam_test.py --mgr 192.168.100.221 --ap1_bssid "68:7d:b4:5f:5c:3b" --ap2_bssid "14:16:9d:53:58:cb"
     --fiveg_radios "1.1.wiphy1" --band "fiveg" --sniff_radio "wiphy2" --num_sta 10 --ssid_name "RoamAP5g" --security "wpa2"
      --security_key "something" --duration None --upstream "eth2" --iteration 10 --channel "40" --option "ota"
      --dut_name ["AP1","AP2"] --traffic_type "lf_udp" --log_file False --debug False --iteration_based --soft_roam True

==============================================================================



optional arguments:
  -h, --help            show this help message and exit
  --help_summary        Show summary of what this script does

Required arguments:
  --mgr MGR             lanforge ip
  --lanforge_port LANFORGE_PORT
                        lanforge port
  --lanforge_ssh_port LANFORGE_SSH_PORT
                        lanforge ssh port
  --ap1_bssid AP1_BSSID
                        AP1 bssid
  --ap2_bssid AP2_BSSID
                        AP2 bssid
  --twog_radios TWOG_RADIOS
                        Twog radio
  --fiveg_radios FIVEG_RADIOS
                        Fiveg radio
  --sixg_radios SIXG_RADIOS
                        Sixg radio
  --band BAND           eg. --band "twog" or sixg
  --sniff_radio SNIFF_RADIO
                        eg. --sniff_radio "wiphy2
  --num_sta NUM_STA     eg. --num_sta 1
  --ssid_name SSID_NAME
                        eg. --ssid_name "ssid_5g"
  --security SECURITY   eg. --security "wpa2"
  --security_key SECURITY_KEY
                        eg. --security_key "something"
  --upstream UPSTREAM   eg. --upstream "eth2"
  --duration DURATION   duration
  --iteration ITERATION
                        Number of iterations
  --channel CHANNEL     Channel
  --option OPTION       eg. --option "ota
  --iteration_based     Iteration based
  --duration_based      Duration based
  --dut_name DUT_NAME
  --traffic_type TRAFFIC_TYPE
                        To chose the traffic type
  --identity IDENTITY   Radius server identity
  --ttls_pass TTLS_PASS
                        Radius Server passwd
  --log_file LOG_FILE   To get the log file, need to pass the True
  --debug DEBUG         To enable/disable debugger, need to pass the True/False
  --soft_roam SOFT_ROAM
                        To enable soft rome eg. --soft_rome True
  --sta_type STA_TYPE   provide the type of client you want to creatE i.e 11r,11r-sae, 11r-sae-802.1x or simple as none
  --multicast MULTICAST
                        set to true only if we want multicast traffic run along the hard roam process

Optional arguments:
  --scheme SCHEME
  --dest DEST
  --user USER
  --passwd PASSWD
  --prompt PROMPT
  --series_cc SERIES_CC
  --ap AP
  --ap_ssh_port AP_SSH_PORT
  --band_cc BAND_CC
  --timeout TIMEOUT

               lf_roam_test.py
```

py-scripts/lf_rssi_process.py

```
usage: lf_rssi_process.py [-h] [--csv CSV] [--png_dir o]
                          [--bandwidths BANDWIDTHS] [--channels CHANNELS]
                          [--antennas ANTENNAS]
                          [--pathloss_list PATHLOSS_LIST]
                          [--log_level LOG_LEVEL]
                          [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                          [--help_summary]

       lf_rssi_process.py:
       --------------------


optional arguments:
  -h, --help            show this help message and exit
  --csv CSV             ../output.csv
```

```
--png_dir o           ../PNGs
--bandwidths BANDWIDTHS
                      --bandwidths  list of bandwidths "20 40 80 160" space separated, default : "20"
--channels CHANNELS   --channels  list of channels "6 36" space separated, default: "36"
--antennas ANTENNAS
                                            --antennas list of antennas "0, 1, 4, 7, 8"  default:
                                                    self.ANTENNA_LEGEND = {
                                                        0: 'Diversity (All)',
                                                        1: 'Fixed-A (1x1)',
                                                        4: 'AB (2x2)',
                                                        7: 'ABC (3x3)',
                                                        8: 'ABCD (4x4)'
                                                    }
                                                    default is 0

--pathloss_list PATHLOSS_LIST
                      list of path loss for 2g, 5g, 6g default: 26.74 31.87 0
--log_level LOG_LEVEL
                      Set logging level: debug | info | warning | error | critical
--lf_logger_config_json LF_LOGGER_CONFIG_JSON
                      --lf_logger_config_json <json file> , json configuration of logger
--help_summary        Show summary of what this script does

          # Exit Codes
          # 0: Success
          # 1: Python Error
          # 2: CSV file not found
          # 3: Radio disconnected before exit threshold expected RSSI; PNG will still be generated
          # 4: Attempted Bandwidth HT80 used with Channel 6
```

py-scripts/lf_rvr_test.py

Example report: rate_vs_range.pdf

```
usage: lf_rvr_test.py [-h] [-m MGR] [-o PORT] [--lf_user LF_USER]
                      [--lf_password LF_PASSWORD] [-i INSTANCE_NAME]
                      [-c CONFIG_NAME] [-r] [--load_old_cfg] [--enable ENABLE]
                      [--disable DISABLE] [--set SET SET]
                      [--raw_line RAW_LINE] [--raw_lines_file RAW_LINES_FILE]
                      [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                      [-u UPSTREAM] [--station STATION]
                      [--band {5g,24g,6g,dual_band_5g,dual_band_6g}]
                      [--radio RADIO] [--create_station] [--ssid SSID]
                      [--ssidpw SSIDPW] [--bssid BSSID] [--security SECURITY]
                      [--wifi_mode WIFI_MODE] [--vht160]
                      [--ieee80211w IEEE80211W] [--dut DUT]
                      [--download_speed DOWNLOAD_SPEED]
                      [--upload_speed UPLOAD_SPEED] [--duration DURATION]
                      [--verbosity VERBOSITY] [--graph_groups GRAPH_GROUPS]
                      [--report_dir REPORT_DIR]
                      [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                      [--log_level LOG_LEVEL]
                      [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                      [--help_summary]

   TEST NAME        :   Rate vs Range Test

   TEST SCRIPT      :   lf_rvr_test.py

   PURPOSE          :   The Purpose of this script is to caluclate the Throughput rate with increasing attenuation with one emula

   NOTES            :   To Run this script gui should be opened with

                            path: cd LANforgeGUI_5.4.3 (5.4.3 can be changed with GUI version)
                            pwd (Output : /home/lanforge/LANforgeGUI_5.4.3)
                            ./lfclient.bash -cli-socket 3990

                            This script is used to automate running Rate-vs-Range tests.  You
                            may need to view a Rate-vs-Range test configured through the GUI to understand
                            the options and how best to input data.
   EXAMPLE-1        :
                            ./lf_rvr_test.py --mgr 192.168.100.205 --lf_user lanforge --lf_password lanforge
                            --instance_name rvr-instance --config_name test_con --upstream 1.2.vap0000
                            --dut routed-AP --duration 1m --station 1.1.sta0000 --download_speed 85%
                            --upload_speed 56Kbps --raw_line 'pkts: MTU' --raw_line 'directions: DUT Transmit'
                            --raw_line 'traffic_types: TCP' --raw_line 'attenuator: 1.1.3219' --raw_line 'attenuations: 0..+50..95
                            --raw_line 'attenuator_mod: 243' --ssid rvr_2g --ssidpw Password@123 --security wpa2 --radio wiphy0
                            --bssid DEFAULT --create_station

   EXAMPLE-2        :

                            ./lf_rvr_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge \
                            --instance_name rvr-instance --config_name test_con --upstream 1.1.eth1 \
                            --dut RootAP --duration 15s --station 1.1.wlan0 \
                            --download_speed 85% --upload_speed 56Kbps \
                            --raw_line 'pkts: MTU' \
                            --raw_line 'directions: DUT Transmit' \
                            --raw_line 'traffic_types: TCP' \
                            --test_rig Ferndale-Mesh-01 --pull_report \
                            --raw_line 'attenuator: 1.1.1040' \
                            --raw_line 'attenuations: 0..+50..950' \
                            --raw_line 'attenuator_mod: 3'

   SCRIPT_CLASSIFICATION:  Test

   SCRIPT_CATEGORIES:      Performance,  Functional,  KPI Generation,  Report Generation

   NOTES:
       attenuator_mod: selects the attenuator modules, bit-field.
       This example uses 3, which is first two attenuator modules on Attenuator ID 1040.

       --raw_line 'line contents' will add any setting to the test config.  This is
           useful way to support any options not specifically enabled by the
           command options.
       --set modifications will be applied after the other config has happened,
           so it can be used to override any other config.
           sel_port-0: 1.1.wlan0
```

```
            show_events: 1
            show_log: 0
            port_sorting: 0
            kpi_id: Rate vs Range
            bg: 0xE0ECF8
            test_rig:
            show_scan: 1
            auto_helper: 0
            skip_2: 0
            skip_5: 0
            skip_5b: 1
            skip_dual: 0
            skip_tri: 1
            selected_dut: RootAP
            duration: 15000
            traffic_port: 1.1.6 wlan0
            upstream_port: 1.1.1 eth1
            path_loss: 10
            speed: 85%
            speed2: 56Kbps
            min_rssi_bound: -150
            max_rssi_bound: 0
            channels: AUTO
            modes: Auto
            pkts: MTU
            spatial_streams: AUTO
            security_options: AUTO
            bandw_options: AUTO
            traffic_types: TCP
            directions: DUT Transmit
            txo_preamble: OFDM
            txo_mcs: 0 CCK, OFDM, HT, VHT
            txo_retries: No Retry
            txo_sgi: OFF
            txo_txpower: 15
            attenuator: 1.1.1040
            attenuator2: 0
            attenuator_mod: 243
            attenuator_mod2: 255
            attenuations: 0..+50..950
            attenuations2: 0..+50..950
            chamber: 0
            tt_deg: 0..+45..359
            cust_pkt_sz:
            show_bar_labels: 1
            show_prcnt_tput: 0
            show_3s: 0
            show_ll_graphs: 0
            show_gp_graphs: 1
            show_lm: 1
            pause_iter: 0
            outer_loop_atten: 0
            show_realtime: 1
            operator:
            mconn: 1
            mpkt: 1000
            tos: 0
            loop_iterations: 1

    STATUS: BETA RELEASE

    VERIFIED_ON:
                12th May 2023
                GUI Version    : 5.4.6
                Kernel Version : 5.19.17+

    LICENSE:
        Free to distribute and modify. LANforge systems must be licensed.
        Copyright 2022 Candela Technologies Inc

    INCLUDE_IN_README: False

    Example of raw text config for Rate-vsRange, to show other possible options:



optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the capacity test?  Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).  See example raw text config for possible options.  M
  --disable DISABLE     Specify options to disable (set value to 0).  See example raw text config for possible options.  May be sp
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for instance:  testbed-01
  -u UPSTREAM, --upstream UPSTREAM
                        Upstream port for wifi capacity test ex. 1.1.eth2
  --station STATION     Station to be used in this test, example: 1.1.sta01500
  --band {5g,24g,6g,dual_band_5g,dual_band_6g}
                        band testing --band 6g
  --radio RADIO         [LANforge station configuration] LANforge radio station created on --radio wiphy0
  --create_station      [LANforge station configuration] create LANforge station at the beginning of the test
  --ssid SSID           [station configuration] station ssid, ssid of station must match the wlan created --ssid 6G-wpa3-AP3
  --ssidpw SSIDPW, --security_key SSIDPW
                        [station configuration]  station security key --ssidpw hello123
```

```
  --bssid BSSID, --ap_bssid BSSID
                        [station configuration]  station AP bssid
  --security SECURITY   [station configuration] security type open wpa wpa2 wpa3
  --wifi_mode WIFI_MODE
                        [station configuration] --wifi_mode auto  types auto|a|abg|abgn|abgnAC|abgnAX|an|anAC|anAX|b|bg|bgn|bgnAC|
  --vht160              [station configuration] --vht160 , Enable VHT160 in lanforge
  --ieee80211w IEEE80211W
                        [station configuration] --ieee80211w 0 (Disabled) 1 (Optional) 2 (Required) (Required needs to be set to R
  --dut DUT             Specify DUT used by this test, example: linksys-8450
  --download_speed DOWNLOAD_SPEED
                        Specify requested download speed.  Percentage of theoretical is also supported.  Default: 85
  --upload_speed UPLOAD_SPEED
                        Specify requested upload speed.  Percentage of theoretical is also supported.  Default: 0
  --duration DURATION   Specify duration of each traffic run
  --verbosity VERBOSITY
                        Specify verbosity of the report values 1 - 11 default 5
  --graph_groups GRAPH_GROUPS
                        File to save graph_groups to
  --report_dir REPORT_DIR
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir <where to pull reports to>  default '' put where dataplane script run from
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --help_summary        Show summary of what this script does
```

py-scripts/lf_rx_sensitivity_test.py

```
usage: lf_rx_sensitivity_test.py [-h] [-m MGR] [-o PORT] [--lf_user LF_USER]
                                 [--lf_password LF_PASSWORD]
                                 [-i INSTANCE_NAME] [-c CONFIG_NAME] [-r]
                                 [--load_old_cfg] [--enable ENABLE]
                                 [--disable DISABLE] [--set SET SET]
                                 [--raw_line RAW_LINE]
                                 [--raw_lines_file RAW_LINES_FILE]
                                 [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                                 [--json JSON] [-u UPSTREAM]
                                 [--station STATION] [--dut DUT]
                                 [--download_speed DOWNLOAD_SPEED]
                                 [--upload_speed UPLOAD_SPEED]
                                 [--duration DURATION]
                                 [--graph_groups GRAPH_GROUPS]
                                 [--report_dir REPORT_DIR] [--help_summary]

    IMPORTANT: Start lanforge with socket 3990 :  ./lfclient.bash -cli-socket 3990
        lfclient.bash is located in the LANforgeGUI_X.X.X directory

        On local or remote system: ./lfclient.bash -cli-socket 3990 -s LF_MGR
        On local system the -s LF_MGR will be local_host if not provided

    Open this file in an editor and read the top notes for more details.
    Example:
    ./lf_rx_sensitivity_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge       --instance_name rx-sen

    Example 2:
    ./lf_rx_sensitivity_test.py --json <name>.json

    see sample json file: lf_rx_sensitivity_config.json

    Sample <name>.json between using eth1 and eth2
    {
     "mgr":"192.168.0.101",
     "port":"8080",
     "lf_user":"lanforge",
     "lf_password":"lanforge",
     "instance_name":"rx-sensitivity-instance",
     "config_name":"test_con",
     "upstream":"1.1.eth1",
     "dut":"asus_5g",
     "duration":"15s",
     "station":"1.1.eth2",
     "download_speed":"85%",
     "upload_speed":"0",
     "raw_line":  ["txo_preamble: VHT", "txo_mcs: 4 OFDM, HT, VHT;5 OFDM, HT, VHT;6 OFDM, HT, VHT;7 OFDM, HT, VHT", "spatial_strea
    }

    Sample <name>.json between using eth1 and station 1.1.sta0002
    {
     "mgr":"192.168.0.101",
     "port":"8080",
     "lf_user":"lanforge",
     "lf_password":"lanforge",
     "instance_name":"rx-sensitivity-instance",
     "config_name":"test_con",
     "upstream":"1.1.eth1",
     "dut":"asus_5g",
     "duration":"15s",
     "station":"1.1.sta0002",
     "download_speed":"85%",
     "upload_speed":"0",
     "raw_line":  ["txo_preamble: VHT", "txo_mcs: 4 OFDM, HT, VHT;5 OFDM, HT, VHT;6 OFDM, HT, VHT;7 OFDM, HT, VHT", "spatial_strea
    }



optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
```

```
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the capacity test?  Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).  See example raw text config for possible options.  M
  --disable DISABLE     Specify options to disable (set value to 0).  See example raw text config for possible options.  May be sp
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for instance:  testbed-01
  --json JSON           --json <config.json> json input file
  -u UPSTREAM, --upstream UPSTREAM
                        Upstream port for rx sensitivity test ex. 1.1.eth2
  --station STATION     Station to be used in this test, example: 1.1.sta01500
  --dut DUT             Specify DUT used by this test, example: linksys-8450
  --download_speed DOWNLOAD_SPEED
                        Specify requested download speed.  Percentage of theoretical is also supported. Default: 85%.
  --upload_speed UPLOAD_SPEED
                        Specify requested upload speed.  Percentage of theoretical is also supported.  Default: 0
  --duration DURATION   Specify duration of each traffic run
  --graph_groups GRAPH_GROUPS
                        File to save graph_groups to
  --report_dir REPORT_DIR
  --help_summary        Show summary of what this script does
```

py-scripts/lf_setup_radius_server.py

```
usage: lf_setup_radius_server.py [-h] [--mgr MGR] [--secret SECRET]
                                 [--input_text INPUT_TEXT] [--help_summary]

lf_setup_radius_server.py: Summary :This script will be able to add a
configuration provided by the user to the Radius server's users configuration.

optional arguments:
  -h, --help            show this help message and exit
  --mgr MGR             ipaddr of connections
  --secret SECRET       secrets for each user
  --input_text INPUT_TEXT
                        client localhost { ipaddr = 10.28.2.0/24 proto = *
                        secret = testing123 require_message_authenticator = no
                        shortname = localhost nas_type = other limit {
                        max_connections = 16 lifetime = 0 idle_timeout = 30 }
                        } client localhost_ipv6 { ipv6addr = ::1 secret =
                        testing123 }
  --help_summary        Show summary of what this script does
```

py-scripts/lf_sniff_radio.py

```
usage:
Creating a sniffer generally:
 ./lf_sniff_radio.py
      --mgr localhost
      --mgr_port 8080
      --radio wiphy0
      --outfile /home/lanforge/test_sniff.pcap
      --duration 1
      --channel 36
      --channel_bw 40
      --center_freq 5190
      --radio_mode AUTO
      --monitor_name moni0

Creating a 6GHz sniffer on AX210/BE200 radios:
 ./lf_sniff_radio.py
      --mgr 192.168.0.104
      --mgr_port 8080
      --radio wiphy7
      --outfile /home/lanforge/sniff_6G_80.pcap
      --duration 20
      //--channel 1e
      --channel_bw 80
      --channel_freq 5955
      --center_freq 5985
      --radio_mode AUTO
      --monitor_name moni0
      --disable_ht40 0
      --disable_ht80 0
      --ht160_enable 0
      --6ghz_workaround
      --num_stations 1
      --security wpa2
      --ssid axe11000_5g
      --password lf_axe11000_5g
      --6ghz_workaround_scan_time 10

        This script will sniff a Radio following modifications to its settings.
         lf_sniff_radio.py will create a monitor and be able to capture wireshark pcap files:

        The monitor also uses iw commands to set up the proper bw and frequency to be monitored

        Note:

        iw [options] dev <devname> set freq <freq> [NOHT|HT20|HT40+|HT40-|5MHz|10MHz|80MHz]
        dev <devname> set freq <control freq> [5|10|20|40|80|80+80|160] [<center1_freq> [<center2_freq>]]

        Example to monitor channel 36 (5180)
        sudo iw dev <monitor/devname> set freq 5180 40 5190

        for bw of 20 do not need to set the control frequency

        Verify the configuration with :(need to do sudo)
        iw dev <monitor/devname> info

        example:
        [lanforge@ct523c-3ba3 ~]$ sudo iw dev SNIFF_5G_40 info
        [sudo] password for lanforge:
        Interface SNIFF_5G_40
```

```
            ifindex 49
            wdev 0x2
            addr d8:f8:83:36:4c:a0
            type monitor
            wiphy 0
            channel 36 (5180 MHz), width: 20 MHz, center1: 5180 MHz
            txpower 0.00 dBm
        [lanforge@ct523c-3ba3 ~]$

        Help: 5Ghz frequencies

        Tested on 02/16/2023:
         kernel version: 5.19.17+
         gui version: 5.4.6
         the script modified a radio and created a pcap file as expected when ran directly on a lanforge system (ct523c & ct521a).



    optional arguments:
      -h, --help            show this help message and exit
      --mgr MGR             IP Address of LANforge
      --mgr_port MGR_PORT   HTTP Port of LANforge
      --radio RADIO         Radio to sniff with
      --outfile OUTFILE     Give the filename with path
      --duration DURATION   Duration in sec for which you want to capture
      --channel CHANNEL     Set channel pn selected Radio, the channel [52, 56 ...]
                            channel will get converted to the control frequency.
                            Must enter Channel
      --channel_freq CHANNEL_FREQ
                            Frequency that the channel operates at
                            Must enter --channel or --channel_freq
                            --channel_freq takes presidence if both entered if value not zero
      --channel_bw CHANNEL_BW
                            Select the bandwidth to be monitored, [ [20|40|80|80+80|160]], default=20
      --center_freq CENTER_FREQ
                            Select the bandwidth to be monitored
                            (not needed if channel width is 20MHz
      --radio_mode RADIO_MODE
                            Select the radio mode [AUTO, 802.11a, 802.11b, 802.11ab ...]
      --monitor_name MONITOR_NAME
                            Wi-Fi monitor name
      --disable_ht40 DISABLE_HT40
                            Enable/Disable "disable_ht40" [0-disable,1-enable]
      --disable_ht80 DISABLE_HT80
                            Enable/Disable "disable_ht80" [0-disable,1-enable]
      --ht160_enable HT160_ENABLE
                            Enable/Disable "ht160_enable\ [0-disable,1-enable]"
      --6ghz_workaround, --ax210
                            Perform workaround for Intel AX210 or BE200 radio 6GHz monitor mode firmware limitation
                            before sniffing packets. Radio firmware requires a scan of 6GHz-capable regulatory domain
                            before granting access to 6GHz channels on a monitor mode interface.
      --6ghz_workaround_scan_time DO_6GHZ_WORKAROUND_SCAN_TIME, --ax210_scan_time DO_6GHZ_WORKAROUND_SCAN_TIME
                            Time to wait for scan in 6GHz workaround
      --num_stations NUM_STATIONS
                            Number of stations to create default 1 for AX210 sniffing
      --number_template NUMBER_TEMPLATE
                            Start the station numbering with a particular number. Default is 0000
      --station_list STATION_LIST [STATION_LIST ...]
                            Optional: User defined station names, can be a comma or space separated list
      --upstream_port UPSTREAM_PORT
                            Upstream port
      --side_a_min_rate SIDE_A_MIN_RATE
                            bps rate minimum for side_a default: 1024000
      --side_b_min_rate SIDE_B_MIN_RATE
                            bps rate minimum for side_b default: 1024000
      --sta_prefix STA_PREFIX
                            Prefix used when creating station
      --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
      --ssid SSID           WiFi SSID for script objects to associate to
      --password PASSWORD   WiFi passphrase/password/key
      --mode MODE           Used to force mode of stations default: 0 (auto)
      --ap AP               Used to force a connection to a particular AP
      --log_level LOG_LEVEL
                            Set logging level: debug | info | warning | error | critical
      --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                            --lf_logger_config_json <json file> , json configuration of logger
      --sniff_bytes SNIFF_BYTES
                            keep this many bytes per packet, helps to reduce overall capture size
      --sniff_using SNIFF_USING
                            Default sniffer is Wireshark, which is only useful from a desktop setting.
                            Combine options with a comma: dumpcap,mate_xterm
                            tshark:             headless tshark utility
                            dumpcap:            headless dumpcap utility
                            mate_terminal:      make tshark/dumpcap interactive in a MATE terminal
                            mate_xterm:         make tshark/dumpcap interactive in an xterm
                            mate_kill_dumpcap:  kill previously issued dumpcap
      --help_summary        shows summary of the script

    lf_sniff_radio.py will create a monitor on LANforge (cli command add_monitor)
```

py-scripts/lf_snp_test.py

```
usage: lf_snp_test.py [-h] [-ct1] [-ct2] [-ct3] [-cca CONTROLLER_APS]
                      [-ccf CONTROLLER_BANDS] [-cwm CONTROLLER_WIFIMODES]
                      [-cc5 CONTROLLER_CHAN_5GHZS]
                      [-cc2 CONTROLLER_CHAN_24GHZS]
                      [-ccw CONTROLLER_CHAN_WIDTHS] [-cam CONTROLLER_AP_MODES]
                      [-pdu CONTROLLER_PDUS]
                      [-cde CONTROLLER_DATA_ENCRYPTIONS] [-cs {9800,3504}]
                      [-ccp CONTROLLER_PROMPT] [-cas CONTROLLER_AP_SLOT]
                      [-cwl CONTROLLER_WLAN] [-cc CONTROLLER_IP]
                      [-cp CONTROLLER_PORT] [-cu CONTROLLER_USER]
                      [-cpw CONTROLLER_PASSWD] [-ccs {serial,telnet,ssh}]
                      [-ccd CONTROLLER_CLIENT_DENSITIES]
                      [-ctp {1,2,3,4,5,6,7,8}] [-cco] [-api AP_INFO] [-lm MGR]
                      [-d TEST_DURATION] [-pi POLLING_INTERVAL] [--tos TOS]
                      [-db] [-t ENDP_TYPES] [-cd CONTROLLER_DIRECTIONS]
```

```
                    [-u UPSTREAM_PORT] [-o CSV_OUTFILE] [-l] [-c CSV_OUTPUT]
                    [-r RADIO] [-ul_bps SIDE_A_TX_MIN_BPS]
                    [-dl_bps SIDE_B_TX_MIN_BPS] [-noc] [-nos]
                    [-wto WAIT_TIMEOUT] [-ptc] [--help_summary]

lf_snp_test.py:
--------------------

################################################################################
Task Description:  Ultimate Aim
################################################################################
-----------------
Candela Scaling and Performance Test (SNP)

The Test supports configuraiton of a Controller which configures
An AP and the Configuration of LANforge or Multiple LANforges
configured into a "Realm".

########################################
# Examples
# ######################################
EXAMPLE:

Use --print_test_config at end of command to see test configuration

Test configurations take presidence to command line parameters

Using Coded Test Configuration --controller_test_1
    ./lf_snp_test.py --controller_ip 10.195.197.234 --controller_user admin --controller_passwd Milpitas@123
    --controller_aps 'Vanc-e' --controller_series "9800" --endp_types 'lf_udp' --upstream_port eth2 --controller_prompt "Can-SnP-9
    --print_test_config

Using Coded Test Configuration --controller_test_1
    ./lf_snp_test.py --controller_ip 10.195.197.234 --controller_user admin --controller_passwd Milpitas@123
    --controller_aps 'Vanc-e' --controller_series "9800" --endp_types 'lf_udp' --upstream_port eth2 --controller_prompt "Can-SnP-9
    --print_test_config

Using Coded Test Configuration:
    ./lf_snp_test.py -cc 192.168.100.112 -cu admin -cpw Cisco123 -cca APA453.0E7B.CF9C -cs "3504" --endp_types 'lf_udp' --upstream
    --controller_prompt "(Cisco Controller)"
    --print_test_config

Using Commandline with defaults:
    ./lf_snp_test.py --controller_ip 192.168.100.112 --controller_user admin --controller_passwd Cisco123 --controller_aps APA453.
    --controller_prompt "(Cisco Controller)" --radio "radio==1.wiphy0 stations==1  ssid==test_candela ssid_pw==[BLANK] security==o
    --print_test_config

Using Commandline:
    ./lf_snp_test.py --controller_ip 192.168.100.112 --controller_user admin --controller_passwd Cisco123 --controller_aps APA453.
    --controller_series "3504" --upstream_port eth2  --controller_prompt "(Cisco Controller)" --controller_wifimode "a" --controll
    --radio "radio==1.wiphy0 stations==10  ssid==test_candela ssid_pw==[BLANK] security==open wifimode==ac" --controller_client_de
    --print_test_config

Using Commandline: Setting --test_duration "20s" --polling_interval to 5s -ccd "2" (--controller_client_densities)
    ./lf_snp_test.py --controller_ip 192.168.100.112 --controller_user admin --controller_passwd Cisco123 --controller_aps APA453.
    --controller_series "3504" --upstream_port eth2  --controller_prompt "(Cisco Controller)" --controller_wifimode "auto"  --cont
    --radio "radio==1.wiphy0 stations==2  ssid==test_candela ssid_pw==[BLANK] security==open wifimode==an" --controller_client_den
    --print_test_config

############################################
############################################
LANforge Information and General Information
############################################
############################################

############################################
Radios Description
############################################

Radio descriptions:
ax200: so if AP is /n, then ax200 will connect at /n.  But if AP is /AX, we have no way to force ax200 to act like /n
ax200: is dual band, supporting at least /b/g/n/AX on 2.4Ghz, and /a/n/ac/AX on 5Ghz.  2.4Ghz doesn't officially support /AC, but

ath10K: if they want /AC or /n or /abg stations, then our ath10k radios can support that need (and ath9k if they have any, can do
ath10K(998x)  - wave -1 , dual band card it can be ac, n , a/b/g modes, up to 3x3 spacial streams
ath10K(9884) - wave-2 supports 4x4  802.11an-AC  5ghz  (can act as ac , an)

Note: wave-2 radios can act as ac, an, (802.11an-AC) or legacy a/b/g (802.11bgn-AC)

############################################
Wifi Modes
############################################
11ax (2.4 ghz or 5 ghz), 11ac (5 ghz only), 11n (2.4ghz or 5 ghz), 11bg (2.4 ghz)  (controller)

############################################
5 Ghz Radios and Wifi Modes
############################################
Wifi mode: 11ax  - 5ghz
Radios   :  ax200  :        802.11 /a/n/ac/AX

Wifi mode: 11ac - 5ghz
Radios   :  ath10K(9984)    802.11an-AC (9984 are single band)

Wifi mode: 11n - 5ghz
Radios   :  ath10K(9984)    802.11an-AC (9984 are single band)

############################################
24 Ghz Radios and Wifi Modes
############################################
Wifi mode: 11ax - 24ghz
Radios   :  ax200 -         802.11 /b/g/n/AX

Wifi mode: 11ac - 24ghz
Radios   :  ax200           802.11 /b/g/n/AX (2.4Ghz doesn't officially support /AC, but often chips will do /AC there anyway) (in

Wifi mode: 11n - 24ghz
Radios   :  ax200           802.11 /b/g/n/AX
```

```
Wifi mode: 11bg - 24ghz
Radios   :  ax200           802.11 /b/g/n/AX

########################################
Radio Mode Configuration
########################################
controller_wifimode == "anAX" or controller_wifimode == "abgn" or controller_wifimode == "bg":
        radios = radio_AX200_abgn_ax_dict[controller_client_density]

controller_wifimode == "an" or controller_wifimode == "anAC":
        radios = radio_ath10K_9984_an_AC_dict[controller_client_density]

########################################
LANforge Realm Configuration
########################################

1.wiphy0  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
1.wiphy1  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
1.wiphy2  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
1.wiphy3  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
1.wiphy4  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
1.wiphy5  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
1.wiphy6  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
1.wiphy7  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
1.wiphy8  802.11an-AC    ath10k(9984)    523 - 64 stations - 5ghz
1.wiphy9  802.11an-AC    ath10k(9984)    523 - 64 stations - 5ghz

2.wiphy0  802.11abgn-ax  iwlwifi(AX200)  521 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
2.wiphy1  802.11abgn-ax  iwlwifi(AX200)  521 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn

3.wiphy0  802.11abgn-ax  iwlwifi(AX200)  521 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
3.wiphy1  802.11abgn-ax  iwlwifi(AX200)  521 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn

4.wiphy0  802.11abgn-ax  iwlwifi(AX200)  521 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
4.wiphy1  802.11abgn-ax  iwlwifi(AX200)  521 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn

5.wiphy0  802.11abgn-ax  iwlwifi(AX200)  521 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
5.wiphy1  802.11abgn-ax  iwlwifi(AX200)  521 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn

6.wiphy0  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
6.wiphy1  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
6.wiphy2  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
6.wiphy3  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
6.wiphy4  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
6.wiphy5  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
6.wiphy6  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
6.wiphy7  802.11abgn-ax  iwlwifi(AX200)  523 - 1  stations - 5ghz/24ghz use only for 802.11ax - 24gz abgn
6.wiphy8  802.11an-AC    ath10k(9984)    523 - 64 stations - 5ghz
6.wiphy9  802.11an-AC    ath10k(9984)    523 - 64 stations - 5ghz

#############################################################################################################
TECHNICAL UNDERSTANDING: LANForge
#############################################################################################################
    LANForge Monitored Values Per Polling Interval
        'rx bytes' - bytes transmitted
        'rx rate'  - bits per second

    in DL direction: -B tx -> -A rx, (side_b_tx_min_bps) LANforge Eth endpoint transmits bytes (AP/DUT), station endpoint (Wifi) L
    in UL direction: -A tx -> -B rx, (side_a_tx_min_bps) LANforge Eth endpoint receives bytes (AP/DUT), station endpoint (Wifi) LA

#############################################################################################################
LANforge GUI what is displayed in the Column and how to access the value with cli or json
#############################################################################################################
# NOTE: see how rx rate is used in script and can monitor any values in similiar manner

    GUI Column Display       Layer3_cols argument to type in (to print in report)

    Name                |  'name'
    EID                 |  'eid'
    Run                 |  'run'
    Mng                 |  'mng'
    Script              |  'script'
    Tx Rate             |  'tx rate'
    Tx Rate (1 min)     |  'tx rate (1 min)'
    Tx Rate (last)      |  'tx rate (last)'
    Tx Rate LL          |  'tx rate ll'
    Rx Rate             |  'rx rate'
    Rx Rate (1 min)     |  'rx rate (1 min)'
    Rx Rate (last)      |  'rx rate (last)'
    Rx Rate LL          |  'rx rate ll'
    Rx Drop %           |  'rx drop %'
    Tx PDUs             |  'tx pdus'
    Tx Pkts LL          |  'tx pkts ll'
    PDU/s TX            |  'pdu/s tx'
    Pps TX LL           |  'pps tx ll'
    Rx PDUs             |  'rx pdus'
    Rx Pkts LL          |  'pps rx ll'
    PDU/s RX            |  'pdu/s tx'
    Pps RX LL           |  'pps rx ll'
    Delay               |  'delay'
    Dropped             |  'dropped'
    Jitter              |  'jitter'
    Tx Bytes            |  'tx bytes'
    Rx Bytes            |  'rx bytes'
    Replays             |  'replays'
    TCP Rtx             |  'tcp rtx'
    Dup Pkts            |  'dup pkts'
    Rx Dup %            |  'rx dup %'
    OOO Pkts            |  'ooo pkts'
    Rx OOO %            |  'rx ooo %'
    RX Wrong Dev        |  'rx wrong dev'
    CRC Fail            |  'crc fail'
    RX BER              |  'rx ber'
    CX Active           |  'cx active'
    CX Estab/s          |  'cx estab/s'
    1st RX              |  '1st rx'
```

```
   CX TO               |  'cx to'
   Pattern             |  'pattern'
   Min PDU             |  'min pdu'
   Max PDU             |  'max pdu'
   Min Rate            |  'min rate'
   Max Rate            |  'max rate'
   Send Buf            |  'send buf'
   Rcv Buf             |  'rcv buf'
   CWND                |  'cwnd'
   TCP MSS             |  'tcp mss'
   Bursty              |  'bursty'
   A/B                 |  'a/b'
   Elapsed             |  'elapsed'
   Destination Addr    |  'destination addr'
   Source Addr         |  'source addr'


optional arguments:
  -h, --help            show this help message and exit
  -ct1, --controller_test_1
                        --controller_test_1 LANforge static radio configuration
  -ct2, --controller_test_2
                        --controller_test_2 LANforge static radio configuration
  -ct3, --controller_test_3
                        --controller_test_3 LANforge static radio configuration
  -cca CONTROLLER_APS, --controller_aps CONTROLLER_APS
                        --controller_aps List of APs to test  default: APA453.0E7B.CF9C
  -ccf CONTROLLER_BANDS, --controller_bands CONTROLLER_BANDS
                        --controller_bands <a | b | abgn> default: a
  -cwm CONTROLLER_WIFIMODES, --controller_wifimodes CONTROLLER_WIFIMODES
                        List of of wifi mode to test default: auto
  -cc5 CONTROLLER_CHAN_5GHZS, --controller_chan_5ghzs CONTROLLER_CHAN_5GHZS
                        --controller_chan_5ghzs <36 40 ...> default 36
  -cc2 CONTROLLER_CHAN_24GHZS, --controller_chan_24ghzs CONTROLLER_CHAN_24GHZS
                        --controller_chan_24ghzs <1 2 ...> default 1
  -ccw CONTROLLER_CHAN_WIDTHS, --controller_chan_widths CONTROLLER_CHAN_WIDTHS
                        --controller_chan_widths <20 40 80 160> default: "20"
  -cam CONTROLLER_AP_MODES, --controller_ap_modes CONTROLLER_AP_MODES
                        --controller_ap_modes <local flexconnect> default local
  -pdu CONTROLLER_PDUS, --controller_pdus CONTROLLER_PDUS
                        --controller_pdus List of packet sizes "88 512 1370 1518" default 1580
  -cde CONTROLLER_DATA_ENCRYPTIONS, --controller_data_encryptions CONTROLLER_DATA_ENCRYPTIONS
                        --controller_data_encryptions "enable disable"
  -cs {9800,3504}, --controller_series {9800,3504}
                        --controller_series <9800 | 3504>
  -ccp CONTROLLER_PROMPT, --controller_prompt CONTROLLER_PROMPT
                        controller prompt default WLC
  -cas CONTROLLER_AP_SLOT, --controller_ap_slot CONTROLLER_AP_SLOT
                        AP slot, default 1
  -cwl CONTROLLER_WLAN, --controller_wlan CONTROLLER_WLAN
                        --controller_wlan <wlan>, default wlan
  -cc CONTROLLER_IP, --controller_ip CONTROLLER_IP
                        --controller_ip <IP of controller Controller> default 192.168.100.178
  -cp CONTROLLER_PORT, --controller_port CONTROLLER_PORT
                        --controller_port <port of controller Controller> ssh default 22
  -cu CONTROLLER_USER, --controller_user CONTROLLER_USER
                        --controller_user <User-name for controller Controller>
  -cpw CONTROLLER_PASSWD, --controller_passwd CONTROLLER_PASSWD
                        --controller_passwd <Password for controller Controller>
  -ccs {serial,telnet,ssh}, --controller_scheme {serial,telnet,ssh}
                        --controller_scheme (serial|telnet|ssh): connect via serial, ssh or telnet
  -ccd CONTROLLER_CLIENT_DENSITIES, --controller_client_densities CONTROLLER_CLIENT_DENSITIES
                        --controller_client_densities List of client densities defaults 1
  -ctp {1,2,3,4,5,6,7,8}, --controller_tx_powers {1,2,3,4,5,6,7,8}
                        --controller_tx_powers <1 | 2 | 3 | 4 | 5 | 6 | 7 | 8>  1 is highest power default 3
  -cco, --cap_ctl_out   --cap_ctl_out , switch the controller controller output will be captured
  -api AP_INFO, --ap_info AP_INFO
                        (enter 0 if does not apply) --ap_info "ap_scheme==<telnet,ssh or serial> ap_prompt==<ap_prompt> ap_ip==<ap
  -lm MGR, --mgr MGR    --mgr <hostname for where LANforge GUI is running>
  -d TEST_DURATION, --test_duration TEST_DURATION
                        --test_duration <how long to run>  example --time 5d (5 days) default: 2m options: number followed by d, h
  -pi POLLING_INTERVAL, --polling_interval POLLING_INTERVAL
                        --polling_interval <seconds>
  --tos TOS             --tos:  Support different ToS settings: BK | BE | VI | VO | numeric
  -db, --debug          --debug:  Enable debugging
  -t ENDP_TYPES, --endp_types ENDP_TYPES
                        --endp_types <types of traffic> example --endp_types "lf_udp lf_tcp"  Default: lf_udp lf_tcp, options: lf_
  -cd CONTROLLER_DIRECTIONS, --controller_directions CONTROLLER_DIRECTIONS
                        --controller_directions <upload download> example --controller_directions "upload download"  Default: uplo
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        --upstream_port <cross connect upstream_port> example: --upstream_port eth1
  -o CSV_OUTFILE, --csv_outfile CSV_OUTFILE
                        --csv_outfile <Output file for csv data>
  -l, --log             create logfile for messages, default stdout
  -c CSV_OUTPUT, --csv_output CSV_OUTPUT
                        Generate csv output
  -r RADIO, --radio RADIO
                        --radio                          "radio==<number_of_wiphy stations=<=number of stations> ssid==<ssid> ssid
  -ul_bps SIDE_A_TX_MIN_BPS, --side_a_tx_min_bps SIDE_A_TX_MIN_BPS
                        --side_a_tx_min_bps , upload (A side tx) min tx rate bps  default 256000 500000000
  -dl_bps SIDE_B_TX_MIN_BPS, --side_b_tx_min_bps SIDE_B_TX_MIN_BPS
                        --side_b_tx_min_bps , download(B side tx) min tx rate bps  default 1000000000
  -noc, --no_controller
                        -noc / --no_controller no configuration of the controller
  -nos, --no_stations   -nos / --no_stations , no stations
  -wto WAIT_TIMEOUT, --wait_timeout WAIT_TIMEOUT
                        -wto / --wait_timeout , time to wait for stations to get IP
  -ptc, --print_test_config
                        -ptc / --print_test_config , print out the test configuration and exit
  --help_summary        Show summary of what this script does

           Scaling and Performance
```

py-scripts/lf_test_generic.py

```
usage: ./lf_test_generic.py [-h] [--lf_user LF_USER] [--lf_passwd LF_PASSWD]
```

```
                        [--test_type TEST_TYPE] [--mgr MGR]
                        [--mgr_port MGR_PORT] [--cmd CMD]
                        [--spdtest_enable_debug] [--spdtest_enable_report]
                        [--spdtest_no_download] [--spdtest_no_upload]
                        [--spdtest_single_connection] [--spdtest_ookla]
                        [--file_output_lfcurl FILE_OUTPUT_LFCURL]
                        [--destination_url_lfcurl DESTINATION_URL_LFCURL]
                        [--loop_count LOOP_COUNT] [--interval INTERVAL]
                        [--target TARGET] [--client_port CLIENT_PORT]
                        [--server_port SERVER_PORT]
                        [--use_existing_eid USE_EXISTING_EID]
                        [--radio RADIO] [--num_stations NUM_STATIONS]
                        [--ssid SSID] [--passwd PASSWD] [--mode MODE]
                        [--ap AP] [--security SECURITY]
                        [--dut_hw_version DUT_HW_VERSION]
                        [--dut_sw_version DUT_SW_VERSION]
                        [--dut_model_num DUT_MODEL_NUM]
                        [--dut_serial_num DUT_SERIAL_NUM]
                        [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                        [--output_format OUTPUT_FORMAT]
                        [--report_file_path REPORT_FILE_PATH]
                        [--gen_tab_cols GEN_TAB_COLS]
                        [--port_mgr_cols PORT_MGR_COLS]
                        [--compared_report COMPARED_REPORT]
                        [--create_report]
                        [--monitor_interval MONITOR_INTERVAL]
                        [--test_duration TEST_DURATION]
                        [--log_level LOG_LEVEL]
                        [--lf_logger_json LF_LOGGER_JSON] [--debug]
                        [--help_summary]

    lf_test_generic.py
    --------------------

        EXAMPLES:
            LFPING :
                ./lf_test_generic.py --mgr 192.168.102.211 --test_type ping --lf_user lanforge --lf_passwd lanforge --num_stat
                            --ssid eero-mesh-lanforge --passwd lanforge --security wpa2 --radio wiphy1 --target www.google.com
                            "4s" --create_report --report_file_path "/home/diptidhond/test_generic_1"
            LFCURL :
                ./lf_test_generic.py --mgr 192.168.102.211 --test_type lfcurl --lf_user lanforge --lf_passwd lanforge --num_st
                            --ssid eero-mesh-lanforge --passwd lanforge --security wpa2 --radio wiphy1  --test_duration 4s --c
                            --report_file_path "/home/lanforge/test_generic_1"
            SPEEDTEST :
                ./lf_test_generic.py --mgr 192.168.102.211 --test_type speedtest --lf_user lanforge --lf_passwd lanforge --num
                            --ssid mesh-lanforge --passwd lanforge --security wpa2 --radio wiphy1  --test_duration 2m --create
                            --report_file_path "/home/lanforge/test_generic_1"

            IPERF3 :
                iperf: create 1 client and 1 server. client is already existing & server is on lanforge.

                    ./lf_test_generic.py --mgr 192.168.102.211 --port 8080 --use_existing_eid "1.1.sta00015,1.1.eth3" --test_t
                        --server_port 5201 --client_port 5201 --target "1.1.eth3" --create_report --port_mgr_cols "alias,rx by

                iperf -- client only: creates 3 client : 2 clients on sta, 1 on eth port (with existing eid):

                    ./lf_test_generic.py --mgr 192.168.102.211 --test_type iperf3-client --lf_user lanforge --lf_passwd lanfor
                        --ssid mesh-lanforge --passwd lanforge --security wpa2 --radio wiphy1 --target 192.168.3.3 --use_exist
                        --create_report --report_file_path "/home/lanforge/iperf3_reports" --port_mgr_cols "alias,rx bytes,tx

                iperf -- server only: creates 3 servers : 2 servers on sta, 1 on eth port (with existing eid):

                    ./lf_test_generic.py --mgr 192.168.102.211 --test_type iperf3-server --lf_user lanforge --lf_passwd lanfor
                        --ssid mesh-lanforge --passwd lanforge --security wpa2 --radio wiphy1 --target --use_existing_eid 1.1.
                        --report_file_path "/home/lanforge/iperf3_reports" --port_mgr_cols "alias,rx bytes,tx bytes"

            Port Mgr Cols available to be reported:
                4way time (us)'
                'activity'
                'alias'
                'anqp time (us)'
                'ap'
                'beacon'
                'bps rx'
                'bps rx ll'
                'bps tx'
                'bps tx ll'
                'bytes rx ll'
                'bytes tx ll'
                'channel'
                'collisions'
                'connections'
                'crypt'
                'cx ago'
                'cx time (us)'
                'device'
                'dhcp (ms)'
                'down'
                'entity id'
                'gateway ip'
                'ip'
                'ipv6 address'
                'ipv6 gateway'
                'key/phrase'
                'login-fail'
                'login-ok'
                'logout-fail'
                'logout-ok'
                'mac'
                'mask'
                'misc'
                'mode'
                'mtu'
                'no cx (us)'
                'noise'
                'parent dev'
                'phantom'
```

```
                'port'
                'port type'
                'pps rx'
                'pps tx'
                'qlen'
                'reset'
                'retry failed'
                'rx bytes'
                'rx crc'
                'rx drop'
                'rx errors'
                'rx fifo'
                'rx frame'
                'rx length'
                'rx miss'
                'rx over'
                'rx pkts'
                'rx-rate'
                'sec'
                'signal'
                'ssid'
                'status'
                'time-stamp'
                'tx abort'
                'tx bytes'
                'tx crr'
                'tx errors'
                'tx fifo'
                'tx hb'
                'tx pkts'
                'tx wind'
                'tx-failed %'
                'tx-rate'
                'wifi retries'

            Generic Tab Cols available to be reported:
                'bps rx'
                'bps tx'
                'command'
                'dropped'
                'eid'
                'elapsed'
                'entity id'
                'last results'
                'name'
                'pdu/s rx'
                'pdu/s tx'
                'rpt timer'
                'rpt#'
                'rx bytes'
                'rx pkts'
                'status'
                'tx bytes'
                'tx pkts'
                'type'


optional arguments:
  -h, --help            show this help message and exit

Arguments that must be defined by user::
  --lf_user LF_USER     user: lanforge
  --lf_passwd LF_PASSWD
                        passwd: lanforge
  --test_type TEST_TYPE
                        type of command to run. Options: ping, iperf3-client, iperf3-server, iperf3, lfcurl

Arguments that do not need to be defined by user::
  --mgr MGR             ip address of lanforge script should be run on. example: 192.168.102.211
  --mgr_port MGR_PORT   port which lanforge is running on, on lanforge machine script should be run on. example: 8080
  --cmd CMD             specifies command to be run by generic type endp
  --spdtest_enable_debug
                        check enable debug box for speedtest cross connect(s)
  --spdtest_enable_report
                        check enable report box for speedtest cross connect(s)
  --spdtest_no_download
                        do not run download for speedtest cross connect
  --spdtest_no_upload   do not run upload for speedtest cross connect
  --spdtest_single_connection
                        run speedtest single connection
  --spdtest_ookla       run ookla speedtest. ookla license must be on machine in order for this test to run.
  --file_output_lfcurl FILE_OUTPUT_LFCURL
                        location to output results of lf_curl, absolute path preferred
  --destination_url_lfcurl DESTINATION_URL_LFCURL
                        destination url for lfcurl
  --loop_count LOOP_COUNT
                        determines the number of loops to use in lf_curl and lfping
  --interval INTERVAL   ping interval configuration
  --target TARGET       Target for lfping (ex: www.google.com). ALSO arg to specify IP address (if server is OFF lanforge, ex: 192
  --client_port CLIENT_PORT
                        the port number of the iperf client endpoint. example: -p 5011
  --server_port SERVER_PORT
                        the port number of the iperf server endpoint. example: -p 5011
  --use_existing_eid USE_EXISTING_EID
                        EID of ports we want to use. Example: '1.1.sta000, 1.1.eth1, 1.1.eth2'
  --radio RADIO         radio that stations should be created on
  --num_stations NUM_STATIONS
                        number of stations that are to be made, defaults to 1
  --ssid SSID           ssid for stations to connect to
  --passwd PASSWD, -p PASSWD
                        password to ssid for stations to connect to
  --mode MODE           Used to force mode of stations
  --ap AP               Used to force a connection to a particular AP, bssid of specific AP
  --security SECURITY   security for station ssids. options: {open | wep | wpa | wpa2 | wpa3}
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
```

```
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number of the device under test
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differentiate the test
  --output_format OUTPUT_FORMAT
                        choose either csv or xlsx. currently xlsx is under construction.
  --report_file_path REPORT_FILE_PATH
                        directory to store results in. example: /home/lanforge/report-data/directory-to-store-results
  --gen_tab_cols GEN_TAB_COLS
                        Columns wished to be monitored from generic endpoint tab. please use list format. examples example: "name,
  --port_mgr_cols PORT_MGR_COLS
                        Columns wished to be monitored from port manager tab. example: "ap,ip,parent dev"
  --compared_report COMPARED_REPORT
                        report path and file which is wished to be compared with new report
  --create_report       specify this flag if test should create report. This means that html, pdf, and csv data is saved and creat
  --monitor_interval MONITOR_INTERVAL
                        frequency of monitors measurements;example: 250ms, 35s, 2h
  --test_duration TEST_DURATION
                        duration of the test eg: 30s, 2m, 4h
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_json LF_LOGGER_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --debug, -d           Enable debugging
  --help_summary        Show summary of what this script does

Create generic endpoints and test for their ability to execute chosen commands
```

py-scripts/lf_test_max_association.py

```
usage: lf_test_max_association.py [-h] [-m MGR] [--port PORT]
                                  [--download_bps DOWNLOAD_BPS]
                                  [--upload_bps UPLOAD_BPS]
                                  [--test_duration TEST_DURATION] [--debug]
                                  [--report_file REPORT_FILE]
                                  [--output_format OUTPUT_FORMAT]
                                  [--prefix PREFIX] [--resource RESOURCE]
                                  [--upstream_port UPSTREAM_PORT]
                                  [--sta_mode STA_MODE]
                                  [--bringup_time BRINGUP_TIME]
                                  [--debug_log DEBUG_LOG]
                                  [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                                  [--monitor_interval MONITOR_INTERVAL]
                                  [--compared_report COMPARED_REPORT]
                                  [--layer3_cols LAYER3_COLS]
                                  [--port_mgr_cols PORT_MGR_COLS]
                                  [--radio6 RADIO6] [--test_rig TEST_RIG]
                                  [--test_tag TEST_TAG]
                                  [--dut_hw_version DUT_HW_VERSION]
                                  [--dut_sw_version DUT_SW_VERSION]
                                  [--dut_model_num DUT_MODEL_NUM]
                                  [--dut_serial_num DUT_SERIAL_NUM]
                                  [--test_priority TEST_PRIORITY]
                                  [--test_id TEST_ID]
                                  [--csv_outfile CSV_OUTFILE]
                                  [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                  [--log_level LOG_LEVEL] [-r RADIO]
                                  [--ssid SSID [SSID ...]]
                                  [--ssid_pw SSID_PW [SSID_PW ...]]
                                  [--security SECURITY [SECURITY ...]]
                                  [--wifi_settings]
                                  [--wifi_mode WIFI_MODE [WIFI_MODE ...]]
                                  [--wifi_enable_flags WIFI_ENABLE_FLAGS [WIFI_ENABLE_FLAGS ...]]
                                  [--reset_port_enable]
                                  [--reset_port_time_min RESET_PORT_TIME_MIN [RESET_PORT_TIME_MIN ...]]
                                  [--reset_port_time_max RESET_PORT_TIME_MAX [RESET_PORT_TIME_MAX ...]]
                                  [--help_summary]

NAME: lf_test_max_association.py

PURPOSE:  [LANforge Unit Test]:-  Create maximum stations per wiphy radio.

          This script will conduct a maximum client overnight test for the ct521a and ct523c systems. The following steps will tak

            - Creates the maximum supported stations per installed radio.
            - Associate the created stations to their prospective SSID's.
            - Create sta-to-eth Layer-3 CX for 9.6Kbps bidirectional overnight maximum-client wifi test.

EXAMPLE:
        # To run the test on specified time duration on all available radios
            [Automatically detect the available radios in lanforge & will create the max supported radios on each radios.]

            ./lf_test_max_association.py --mgr 192.168.200.64 --ssid 208NETWORK --ssid_pw lanforge --security wpa2
                --csv_outfile ./lf_test_max_association --test_rig CT_01 --test_tag MAX_STA --dut_hw_version 1.0
                --dut_model_num ct521a--dut_sw_version 5.4.6 --dut_serial_num 361c --upstream_port 1.1.eth1
                --upload_bps 6200000 --download_bps 6200000 --test_duration 30s

        # To run the test based on specified time duration and radios

            ./lf_test_max_association.py --mgr 192.168.200.64
                --radio 'radio==1.1.wiphy0,ssid==208NETWORK,ssid_pw==lanforge,security==wpa2'
                --radio 'radio==1.1.wiphy1,ssid==208NETWORK,ssid_pw==lanforge,security==wpa2'
                --csv_outfile ./lf_test_max_association --test_rig CT_01 --test_tag MAX_STA
                --dut_hw_version 1.0 --dut_model_num ct521a --dut_sw_version 5.4.6 --dut_serial_num 361c
                --test_duration 30s --upstream_port 1.1.eth1

        # To run overnight test using chambered AP's on specified radios only.

            ./lf_test_max_association.py --mgr <localhost>
                --radio 'radio==1.1.wiphy0,ssid==<ssid>,ssid_pw==<password>,security==<type>'
                --radio 'radio==1.1.wiphy1,ssid==<ssid>,ssid_pw==<password>,security==<type>'
                --csv_outfile lf_test_max_association --test_rig CT_01 --test_tag MAX_STA
                --dut_hw_version 1.0 --dut_model_num lf0350 --dut_sw_version 5.4.5 --dut_serial_num 361c
```

```
       # For an overnight test using chambered AP's with no security enabled:
           ./lf_test_max_association.py --mgr <localhost>
               --radio 'radio==1.1.wiphy0,ssid==<ssid>,ssid_pw==[BLANK],security==open'
               --radio 'radio==1.1.wiphy1,ssid==<ssid>,ssid_pw==[BLANK],security==open'
               --csv_outfile lf_test_max_association --test_rig CT_01 --test_tag MAX_STA
               --dut_hw_version 1.0 --dut_model_num lf0350 --dut_sw_version 5.4.5 --dut_serial_num 361c

       # For an overnight test with a ct523c system:
           ./lf_max_association.py --mgr <localhost>
               --radio 'radio==1.1.wiphy0,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy1,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy2,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy3,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy4,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy5,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy6,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy7,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --csv_outfile lf_test_max_association --test_rig CT_01 --test_tag MAX_STA
               --dut_hw_version 1.0 --dut_model_num ct523c --dut_sw_version 5.4.5 --dut_serial_num cc34

       # For an overnight test with a ct523c system and a 6e network:
           ./lf_max_association.py --mgr <localhost>
               --radio 'radio==1.1.wiphy0,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy1,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy2,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy3,ssid==<ssid>,ssid_pw==<password>,security==wpa2'
               --radio 'radio==1.1.wiphy4,ssid==<ssid>,ssid_pw==<password>,security==wpa3'
               --radio 'radio==1.1.wiphy5,ssid==<ssid>,ssid_pw==<password>,security==wpa3'
               --radio 'radio==1.1.wiphy6,ssid==<ssid>,ssid_pw==<password>,security==wpa3'
               --radio 'radio==1.1.wiphy7,ssid==<ssid>,ssid_pw==<password>,security==wpa3'
               --radio6 1.1.wiphy4 --radio6 1.1.wiphy5 --radio6 1.1.wiphy6 --radio6 1.1.wiphy7
               --csv_outfile lf_test_max_association --test_rig CT_01 --test_tag MAX_STA
               --dut_hw_version 1.0 --dut_model_num ct523c --dut_sw_version 5.4.5 --dut_serial_num cc34

SCRIPT_CLASSIFICATION:  Traffic Test, Stations & Cross-connectionCreation, Report generation

SCRIPT_CATEGORIES:   Functional

NOTES:  This script serves two main purposes:
           * It can create the maximum supported stations and cross-connections based on the available radio capability.
           * You can choose to create stations on a specific radio using the '--radio' argument, or if on all radios
             no '--radio' argument is provided.
           * Additionally, you have the flexibility to run the script for both short and long durations using the
             '--test_duration' argument."

STATUS: Functional

VERIFIED_ON:   25-AUG-2023,
          Build Version:  5.4.7
          Kernel Version: 6.2.16+

LICENSE:
       Free to distribute and modify. LANforge systems must be licensed.
       Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False

optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  --port PORT           LANforge Manager port
  --download_bps DOWNLOAD_BPS
                        Set the minimum bps value on test endpoint A. Default: 9.6Kbps
  --upload_bps UPLOAD_BPS
                        Set the minimum bps value on test endpoint B. Default: 9.6Kbps
  --test_duration TEST_DURATION
                        --test_duration sets the duration of the test, default is 8 hours
  --debug               enable debugging
  --report_file REPORT_FILE
                        where you want to store results
  --output_format OUTPUT_FORMAT
                        choose either csv or xlsx
  --prefix PREFIX       Station prefix. Default: 'sta'
  --resource RESOURCE   LANforge Station resource ID to use, default is 1
  --upstream_port UPSTREAM_PORT
                        LANforge Ethernet port name, default is eth1
  --sta_mode STA_MODE   LANforge station-mode setting (see add_sta LANforge CLI documentation, default is 0 (auto))
  --bringup_time BRINGUP_TIME
                        Seconds to wait for stations to associate and aquire IP. Default: 300
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path, primary use when running test in test suite
  --monitor_interval MONITOR_INTERVAL
                        how frequently do you want your monitor function to take measurements, 35s, 2h
  --compared_report COMPARED_REPORT
                        report path and file which is wished to be compared with new report
  --layer3_cols LAYER3_COLS
                        Columns wished to be monitored from layer 3 endpoint tab
  --port_mgr_cols PORT_MGR_COLS
                        Columns wished to be monitored from port manager tab
  --radio6 RADIO6       Specify 6Ghz radio.  May be specified multiple times.
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differenciate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number / serial number of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --test_id TEST_ID     test-id for kpi.csv,  script or test name
  --csv_outfile CSV_OUTFILE
                        --csv_outfile <prepend input to generated file name for csv data>
```

```
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  -r RADIO, --radio RADIO
                         --radio "radio==<number_of_wiphy ssid==<ssid> ssid_pw==<ssid password> security==<security>  wifi_setting
  --ssid SSID [SSID ...]
                        Enter ssid for all stations with respect to the radios
  --ssid_pw SSID_PW [SSID_PW ...]
                        Enter password for all stations with respect to the radios
  --security SECURITY [SECURITY ...]
                        Enter security for all stations
  --wifi_settings       To enable wifi settings
  --wifi_mode WIFI_MODE [WIFI_MODE ...]
                        Enter mode for stations
  --wifi_enable_flags WIFI_ENABLE_FLAGS [WIFI_ENABLE_FLAGS ...]
                        Enter flags need to be enable
  --reset_port_enable   To enable reset port settings
  --reset_port_time_min RESET_PORT_TIME_MIN [RESET_PORT_TIME_MIN ...]
                        Enter minimum port reset time
  --reset_port_time_max RESET_PORT_TIME_MAX [RESET_PORT_TIME_MAX ...]
                        Enter maximum port reset time
  --help_summary        Show summary of what this script does
```

py-scripts/lf_tr398_test.py

```
usage: lf_tr398_test.py [-h] [-m MGR] [-o PORT] [--lf_user LF_USER]
                        [--lf_password LF_PASSWORD] [-i INSTANCE_NAME]
                        [-c CONFIG_NAME] [-r] [--load_old_cfg]
                        [--enable ENABLE] [--disable DISABLE] [--set SET SET]
                        [--raw_line RAW_LINE]
                        [--raw_lines_file RAW_LINES_FILE]
                        [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                        [-u UPSTREAM] [--dut2 DUT2] [--dut5 DUT5]
                        [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                        [--help_summary]

    Open this file in an editor and read the top notes for more details.

    Example:

  ./lf_tr398_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge \
      --instance_name tr398-instance --config_name test_con \
      --upstream 1.2.eth2 \
      --test_rig Testbed-01 --pull_report \
      --local_lf_report_dir /tmp/my-report \
      --dut5 'TR398-DUT-r750 ruckus-r750-5g 4c:b1:cd:18:e8:ec (1)' \
      --dut2 'TR398-DUT-r750 ruckus-r750-2g 4c:b1:cd:18:e8:e8 (2)' \
      --raw_lines_file cv_examples/example_cfgs/tr398-ferndale-ac-cfg.txt \
      --set 'Calibrate Attenuators' 0 \
      --set 'Receiver Sensitivity' 0 \
      --set 'Maximum Connection' 1 \
      --set 'Maximum Throughput' 1 \
      --set 'Airtime Fairness' 0 \
      --set 'Range Versus Rate' 0 \
      --set 'Spatial Consistency' 0 \
      --set 'Multiple STAs Performance' 0 \
      --set 'Multiple Assoc Stability' 0 \
      --set 'Downlink MU-MIMO' 0 \
      --set 'AP Coexistence' 0 \
      --set 'Long Term Stability' 0

    The contents of the 'raw_lines_file' argument can be obtained by manually configuring the
    TR398 test in the LANforge GUI, then select 'Show Config' on the Advanced configuration tab,
    select that config text, and paste it into a file.  That file is the argument to the
    --raw_lines_file argument.



optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the capacity test?  Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).  See example raw text config for possible options.  M
  --disable DISABLE     Specify options to disable (set value to 0).  See example raw text config for possible options.  May be sp
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for instance:  testbed-01
  -u UPSTREAM, --upstream UPSTREAM
                        Upstream port for wifi capacity test ex. 1.1.eth2
  --dut2 DUT2           Specify 2Ghz DUT used by this test, example: 'TR398-DUT-r750 ruckus-r750-2g 4c:b1:cd:18:e8:e8 (2)'
  --dut5 DUT5           Specify 5Ghz DUT used by this test, example: 'TR398-DUT-r750 ruckus-r750-5g 4c:b1:cd:18:e8:ec (1)'
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir <where to pull reports to>  default '' means put in current working directory
  --help_summary        Show summary of what this script does
```

py-scripts/lf_tr398v2_test.py

```
usage:
    Open this file in an editor and read the top notes for more details.

    Example:

  ./lf_tr398v2_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge \
      --instance_name tr398-instance --config_name test_con \
```

```
    --upstream 1.2.eth2 \
    --test_rig Testbed-01 --pull_report \
    --local_lf_report_dir /tmp/my-report \
    --dut5 'TR398-DUT-r750 ruckus-r750-5g 4c:b1:cd:18:e8:ec (1)' \
    --dut2 'TR398-DUT-r750 ruckus-r750-2g 4c:b1:cd:18:e8:e8 (2)' \
    --raw_lines_file cv_examples/example_cfgs/tr398v2-ferndale-ac-cfg.txt \
    --set 'Calibrate 802.11AX Attenuators' 0 \
    --set 'Calibrate 802.11AC Attenuators' 0 \
    --set '6.1.1 Receiver Sensitivity' 0 \
    --set '6.2.1 Maximum Connection' 0 \
    --set '6.2.2 Maximum Throughput' 1 \
    --set '6.2.3 Airtime Fairness' 0 \
    --set '6.2.4 Dual-Band Throughput' 0 \
    --set '6.2.5 Bi-Directional Throughput' 0 \
    --set '6.3.1 Range Versus Rate' 0 \
    --set '6.3.2 Spatial Consistency' 0 \
    --set '6.3.3 AX Peak Performance' 0 \
    --set '6.4.1 Multiple STAs Performance' 0 \
    --set '6.4.2 Multiple Assoc Stability' 0 \
    --set '6.4.3 Downlink MU-MIMO' 0 \
    --set '6.5.2 AP Coexistence' 0 \
    --set '6.5.1 Long Term Stability' 0

./lf_tr398v2_test.py --mgr 192.168.100.105 --port 8080 --lf_user lanforge\
  --lf_password lanforge --instance_name x \
  --config_name testing --pull_report \
  --local_lf_report_dir /tmp --dut5 'ASUS_70 ASUS_70 f0:2f:74:7c:a5:70 (1)' \
  --dut2 'ASUS_70 ASUS_70 f0:2f:74:7c:a5:70 (1)' --raw_line "upstream_port: 1.1.eth2"


The contents of the 'raw_lines_file' argument can be obtained by manually configuring the
TR398 issue 2 test in the LANforge GUI, then selecting 'Show Config' in the Advanced configuration tab, then
highlighting and pasting that text into file.  That file is the argument to the --raw_lines_file parameter.

Each TR398 test's setting values can be specified by the python script in multiple ways.
For example, each test needs an upstream port. The python script can specify upstream port in several ways
and below is the hierarchy of which upstream port will be the final one in the settings.
 1. --upstream_port argument
 2. --raw_lines argument in the command line
 3. upstream port specified in the --raw_lines_file file.txt
 4. upsteam port loaded from the --config argument


    [-h] [-m MGR] [-o PORT] [--lf_user LF_USER] [--lf_password LF_PASSWORD]
    [-i INSTANCE_NAME] [-c CONFIG_NAME] [-r] [--load_old_cfg]
    [--enable ENABLE] [--disable DISABLE] [--set SET SET]
    [--raw_line RAW_LINE] [--raw_lines_file RAW_LINES_FILE]
    [--test_rig TEST_RIG] [--test_tag TEST_TAG] [-u UPSTREAM] [--dut2 DUT2]
    [--dut5 DUT5] [--dut6 DUT6] [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
    [--help_summary]

optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is
                        default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is
                        default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the
                        capacity test? Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).
                        See example raw text config for possible options. May
                        be specified multiple times. Most tests are enabled by
                        default, except: longterm
  --disable DISABLE     Specify options to disable (set value to 0). See
                        example raw text config for possible options. May be
                        specified multiple times.
  --set SET SET         Specify options to set values based on their label in
                        the GUI. Example: --set 'Basic Client Connectivity' 1
                        May be specified multiple times.
  --raw_line RAW_LINE   Specify lines of the raw config file. Example:
                        --raw_line 'test_rig: Ferndale-01-Basic' See example
                        raw text config for possible options. This is catch-
                        all for any options not available to be specified
                        elsewhere. May be specified multiple times.
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for
                        instance: testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for
                        instance: testbed-01
  -u UPSTREAM, --upstream UPSTREAM
                        Upstream port for wifi capacity test ex. 1.1.eth2
  --dut2 DUT2           Specify 2Ghz DUT used by this test, example:
                        'TR398-DUT-r750 ruckus-r750-2g 4c:b1:cd:18:e8:e8 (2)'
  --dut5 DUT5           Specify 5Ghz DUT used by this test, example:
                        'TR398-DUT-r750 ruckus-r750-5g 4c:b1:cd:18:e8:ec (1)'
  --dut6 DUT6           Specify 6Ghz DUT used by this test, example:
                        'TR398-DUT-r750 ruckus-r750-6g 4c:b1:cd:18:e8:eb (3)'
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir <where to pull reports to>
                        default '' means put in current working directory
  --help_summary        Show summary of what this script does
```

py-scripts/lf_tr398v4_test.py

```
usage:
    Open this file in an editor and read the top notes for more details.

    Example:
```

```
./lf_tr398v4_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge \
    --instance_name tr398-instance --config_name test_con \
    --upstream 1.2.eth2 \
    --test_rig Testbed-01 --pull_report \
    --local_lf_report_dir /tmp/my-report \
    --dut6 'TR398-DUT-r750 ruckus-r750-5g 4c:b1:cd:18:e8:eb (3)' \
    --dut5 'TR398-DUT-r750 ruckus-r750-5g 4c:b1:cd:18:e8:ec (1)' \
    --dut2 'TR398-DUT-r750 ruckus-r750-2g 4c:b1:cd:18:e8:e8 (2)' \
    --raw_lines_file cv_examples/example_cfgs/tr398v4-ferndale-be-cfg.txt \
    --set 'Calibrate 802.11AX Attenuators' 0 \
    --set 'Calibrate 802.11AC Attenuators' 0 \
    --set '6.1.1 Receiver Sensitivity' 0 \
    --set '6.2.1 Maximum Connection' 0 \
    --set '6.2.2 Maximum Throughput' 1 \
    --set '6.2.3 Airtime Fairness' 0 \
    --set '6.2.4 Dual-Band Throughput' 0 \
    --set '6.2.5 Bi-Directional Throughput' 0 \
    --set '6.3.1 Range Versus Rate' 0 \
    --set '6.3.2 Spatial Consistency' 0 \
    --set '6.3.3 AX Peak Performance' 0 \
    --set '6.4.1 Multiple STAs Performance' 0 \
    --set '6.4.2 Multiple Assoc Stability' 0 \
    --set '6.4.3 Downlink MU-MIMO' 0 \
    --set '6.5.2 AP Coexistence' 0 \
    --set '6.5.1 Long Term Stability' 0

./lf_tr398v4_test.py --mgr 192.168.100.105 --port 8080 --lf_user lanforge\
  --lf_password lanforge --instance_name x \
  --config_name testing --pull_report \
  --local_lf_report_dir /tmp --dut5 'ASUS_70 ASUS_70 f0:2f:74:7c:a5:70 (1)' \
  --dut2 'ASUS_70 ASUS_70 f0:2f:74:7c:a5:70 (1)' --raw_line "upstream_port: 1.1.eth2"


 The contents of the 'raw_lines_file' argument can be obtained by manually configuring the
 TR398 issue 2 test in the LANforge GUI, then selecting 'Show Config' in the Advanced configuration tab, then
 highlighting and pasting that text into file.  That file is the argument to the --raw_lines_file parameter.

 Each TR398 test's setting values can be specified by the python script in multiple ways.
 For example, each test needs an upstream port. The python script can specify upstream port in several ways
 and below is the hierarchy of which upstream port will be the final one in the settings.
  1. --upstream_port argument
  2. --raw_lines argument in the command line
  3. upstream port specified in the --raw_lines_file file.txt
  4. upsteam port loaded from the --config argument


      [-h] [-m MGR] [-o PORT] [--lf_user LF_USER] [--lf_password LF_PASSWORD]
      [-i INSTANCE_NAME] [-c CONFIG_NAME] [-r] [--load_old_cfg]
      [--enable ENABLE] [--disable DISABLE] [--set SET SET]
      [--raw_line RAW_LINE] [--raw_lines_file RAW_LINES_FILE]
      [--test_rig TEST_RIG] [--test_tag TEST_TAG] [-u UPSTREAM] [--dut2 DUT2]
      [--dut5 DUT5] [--dut6 DUT6] [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
      [--help_summary]

optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is
                        default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is
                        default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the
                        capacity test? Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).
                        See example raw text config for possible options. May
                        be specified multiple times. Most tests are enabled by
                        default, except: longterm
  --disable DISABLE     Specify options to disable (set value to 0). See
                        example raw text config for possible options. May be
                        specified multiple times.
  --set SET SET         Specify options to set values based on their label in
                        the GUI. Example: --set 'Basic Client Connectivity' 1
                        May be specified multiple times.
  --raw_line RAW_LINE   Specify lines of the raw config file. Example:
                        --raw_line 'test_rig: Ferndale-01-Basic' See example
                        raw text config for possible options. This is catch-
                        all for any options not available to be specified
                        elsewhere. May be specified multiple times.
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for
                        instance: testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for
                        instance: testbed-01
  -u UPSTREAM, --upstream UPSTREAM
                        Upstream port for wifi capacity test ex. 1.1.eth2
  --dut2 DUT2           Specify 2Ghz DUT used by this test, example:
                        'TR398-DUT-r750 ruckus-r750-2g 4c:b1:cd:18:e8:e8 (2)'
  --dut5 DUT5           Specify 5Ghz DUT used by this test, example:
                        'TR398-DUT-r750 ruckus-r750-5g 4c:b1:cd:18:e8:ec (1)'
  --dut6 DUT6           Specify 6Ghz DUT used by this test, example:
                        'TR398-DUT-r750 ruckus-r750-6g 4c:b1:cd:18:e8:eb (3)'
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir <where to pull reports to>
                        default '' means put in current working directory
  --help_summary        Show summary of what this script does
```

```
usage: lf_webpage.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                     [--upstream_port UPSTREAM_PORT]
                     [--num_stations NUM_STATIONS] [--twog_radio TWOG_RADIO]
                     [--fiveg_radio FIVEG_RADIO] [--sixg_radio SIXG_RADIO]
                     [--twog_security TWOG_SECURITY] [--twog_ssid TWOG_SSID]
                     [--twog_passwd TWOG_PASSWD]
                     [--fiveg_security FIVEG_SECURITY]
                     [--fiveg_ssid FIVEG_SSID] [--fiveg_passwd FIVEG_PASSWD]
                     [--sixg_security SIXG_SECURITY] [--sixg_ssid SIXG_SSID]
                     [--sixg_passwd SIXG_PASSWD]
                     [--target_per_ten TARGET_PER_TEN] [--file_size FILE_SIZE]
                     [--bands BANDS [BANDS ...]] [--duration DURATION]
                     [--client_type CLIENT_TYPE] [--threshold_5g THRESHOLD_5G]
                     [--threshold_2g THRESHOLD_2G]
                     [--threshold_both THRESHOLD_BOTH] [--ap_name AP_NAME]
                     [--lf_username LF_USERNAME] [--lf_password LF_PASSWORD]
                     [--ssh_port SSH_PORT] [--test_rig TEST_RIG]
                     [--test_tag TEST_TAG] [--dut_hw_version DUT_HW_VERSION]
                     [--dut_sw_version DUT_SW_VERSION]
                     [--dut_model_num DUT_MODEL_NUM]
                     [--dut_serial_num DUT_SERIAL_NUM]
                     [--test_priority TEST_PRIORITY] [--test_id TEST_ID]
                     [--csv_outfile CSV_OUTFILE] [--dowebgui DOWEBGUI]
                     [--result_dir RESULT_DIR]
                     [--web_ui_device_list DEVICE_LIST]
                     [--test_name TEST_NAME] [--get_url_from_file]
                     [--file_path FILE_PATH] [--help_summary]

   NAME: lf_webpage.py

   PURPOSE:
   lf_webpage.py will verify that N clients are connected on a specified band and can download
   some amount of file data from the HTTP server while measuring the time taken by clients to download the file and number of
   times the file is downloaded.

   EXAMPLE-1:
   Command Line Interface to run download scenario for Real clients
   python3 lf_webpage.py --ap_name "Cisco" --mgr 192.168.200.165 --ssid Cisco-5g --security wpa2 --passwd sharedsecret
   --upstream_port eth1 --duration 10m --bands 5G --client_type Real --file_size 2MB

   EXAMPLE-2:
   Command Line Interface to run download scenario on 5GHz band for Virtual clients
   python3 lf_webpage.py --ap_name "Cisco" --mgr 192.168.200.165 --fiveg_ssid Cisco-5g --fiveg_security wpa2 --fiveg_passwd share
   --fiveg_radio wiphy0 --upstream_port eth1 --duration 1h --bands 5G --client_type Virtual --file_size 2MB --num_stations 3

   EXAMPLE-3:
   Command Line Interface to run download scenario on 2.4GHz band for Virtual clients
   python3 lf_webpage.py --ap_name "Cisco" --mgr 192.168.200.165 --twog_ssid Cisco-2g --twog_security wpa2 --twog_passwd sharedse
   --twog_radio wiphy0 --upstream_port eth1 --duration 1h --bands 2.4G --client_type Virtual --file_size 2MB --num_stations 3

   EXAMPLE-4:
   Command Line Interface to run download scenario on 6GHz band for Virtual clients
   python3 lf_webpage.py --ap_name "Cisco" --mgr 192.168.200.165 --sixg_ssid Cisco-6g --sixg_security wpa3 --sixg_passwd sharedse
   --sixg_radio wiphy0 --upstream_port eth1 --duration 1h --bands 6G --client_type Virtual --file_size 2MB --num_stations 3

   EXAMPLE-5:
   Command Line Interface to run the download or upload HTTP urls from a file using Virtual clients on 5GHz band
   Note: The file should contain the the URLs in following mentioned formate
           formate : "dl https://google.com /dev/null"

   EXAMPLE-6:
   Command Line Interface to run download scenario for Real clients with device list
   python3 lf_webpage.py --ap_name "Cisco" --mgr 192.168.214.219 --fiveg_ssid Cisco-5g --fiveg_security wpa2 --fiveg_passwd share
   --duration 1m --bands 5G --client_type Real --file_size 2MB --device_list 1.12,1.22

   Verified CLI:
   python3 lf_webpage.py --ap_name "Netgear1234" --mgr 192.168.200.38 --fiveg_ssid NETGEAR_5G --fiveg_security wpa2
   --fiveg_passwd Password@123 --fiveg_radio 1.1.wiphy1 --upstream_port 1.1.eth2 --duration 1m --bands 5G
   --client_type Virtual --num_stations 5 --get_url_from_file --file_path /home/lanforge/Desktop/dummy.txt

   SCRIPT_CLASSIFICATION : Test

   SCRIPT_CATEGORIES:   Performance,  Functional,  Report Generation

   NOTES:
   1.Please enter the duration in s,m,h (seconds or minutes or hours).Eg: 30s,5m,48h.
   2.After passing cli, a list will be displayed on terminal which contains available resources to run test.
   The following sentence will be displayed
   Enter the desired resources to run the test:
   Please enter the port numbers seperated by commas ','.
   Example:
   Enter the desired resources to run the test:1.10,1.11,1.12,1.13,1.202,1.203,1.303

   STATUS : Functional

   VERIFIED_ON:
   26-JULY-2024,
   GUI Version:  5.4.8
   Kernel Version: 6.2.16+

   LICENSE :
   Copyright 2023 Candela Technologies Inc
   Free to distribute and modify. LANforge systems must be licensed.

   INCLUDE_IN_README: False



optional arguments:
  -h, --help            show this help message and exit

Required arguments to run lf_webpage.py:
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: eg: eth1
  --file_size FILE_SIZE
```

```
                        specify the size of file you want to download
  --bands BANDS [BANDS ...]
                        specify which band testing you want to run eg 5G, 2.4G, 6G
  --duration DURATION   Please enter the duration in s,m,h (seconds or minutes or hours).Eg: 30s,5m,48h
  --client_type CLIENT_TYPE
                        Enter the type of client. Example:"Real","Virtual"
  --ap_name AP_NAME     specify the ap model
  --dowebgui DOWEBGUI   If true will execute script for webgui

Optional arguments to run lf_webpage.py:
  --num_stations NUM_STATIONS
                        number of stations to create for virtual clients
  --twog_radio TWOG_RADIO
                        specify radio for 2.4G clients
  --fiveg_radio FIVEG_RADIO
                        specify radio for 5 GHz client
  --sixg_radio SIXG_RADIO
                        Specify radio for 6GHz client
  --twog_security TWOG_SECURITY
                        WiFi Security protocol: {open|wep|wpa2|wpa3} for 2.4G clients
  --twog_ssid TWOG_SSID
                        WiFi SSID for script object to associate for 2.4G clients
  --twog_passwd TWOG_PASSWD
                        WiFi passphrase/password/key for 2.4G clients
  --fiveg_security FIVEG_SECURITY
                        WiFi Security protocol: {open|wep|wpa2|wpa3} for 5G clients
  --fiveg_ssid FIVEG_SSID
                        WiFi SSID for script object to associate for 5G clients
  --fiveg_passwd FIVEG_PASSWD
                        WiFi passphrase/password/key for 5G clients
  --sixg_security SIXG_SECURITY
                        WiFi Security protocol: {open|wep|wpa2|wpa3} for 2.4G clients
  --sixg_ssid SIXG_SSID
                        WiFi SSID for script object to associate for 2.4G clients
  --sixg_passwd SIXG_PASSWD
                        WiFi passphrase/password/key for 2.4G clients
  --target_per_ten TARGET_PER_TEN
                        number of request per 10 minutes
  --threshold_5g THRESHOLD_5G
                        Enter the threshold value for 5G Pass/Fail criteria
  --threshold_2g THRESHOLD_2G
                        Enter the threshold value for 2.4G Pass/Fail criteria
  --threshold_both THRESHOLD_BOTH
                        Enter the threshold value for Both Pass/Fail criteria
  --lf_username LF_USERNAME
                        Enter the lanforge user name. Example : 'lanforge'
  --lf_password LF_PASSWORD
                        Enter the lanforge password. Example : 'lanforge'
  --ssh_port SSH_PORT   specify the ssh port eg 22
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differentiate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number / serial number of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --test_id TEST_ID     test-id for kpi.csv,  script or test name
  --csv_outfile CSV_OUTFILE
                        --csv_outfile <Output file for csv data>
  --result_dir RESULT_DIR
                        Specify the result dir to store the runtime logs <Do not use in CLI, --used by webui>
  --web_ui_device_list DEVICE_LIST, --device_list DEVICE_LIST
                        Enter the devices on which the test should be run <Do not use in CLI,--used by webui>
  --test_name TEST_NAME
                        Specify test name to store the runtime csv results <Do not use in CLI, --used by webui>
  --get_url_from_file   Specify to enable the get url from file flag for cx
  --file_path FILE_PATH
                        Specify the path of the file, which has the URLs to download or upload the URLs
  --help_summary        Show summary of what this script does
```

py-scripts/lf_we_can_wifi_capacity.py

```
usage: lf_we_can_wifi_capacity.py [-h] [--mgr MGR] [--port PORT]
                                  [--upstream UPSTREAM]
                                  [--batch_size BATCH_SIZE]
                                  [--protocol PROTOCOL] [--lf_user LF_USER]
                                  [--lf_password LF_PASSWORD]
                                  [--duration DURATION]
                                  [--download_rate DOWNLOAD_RATE]
                                  [--upload_rate UPLOAD_RATE]
                                  [--dut_model DUT_MODEL]
                                  [--ssid_dut_2g SSID_DUT_2G]
                                  [--ssid_dut_5g SSID_DUT_5G]
                                  [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                                  [-s STATIONS] [--help_summary]

Netgear AP DFS Test Script

optional arguments:
  -h, --help            show this help message and exit
  --mgr MGR             host name
  --port PORT           port number
  --upstream UPSTREAM   Upstream port for wifi capacity test ex. 1.1.eth1
  --batch_size BATCH_SIZE
                        station increment ex. 1,2,3
  --protocol PROTOCOL   Protocol ex.TCP-IPv4
  --lf_user LF_USER     lanforge user name ex. root,lanforge
  --lf_password LF_PASSWORD
                        lanforge user password ex. root,lanforge
  --duration DURATION   duration in ms. ex. 5000
  --download_rate DOWNLOAD_RATE
                        Select requested download rate. Kbps, Mbps, Gbps units
```

```
                        supported. Default is 0Kbps
  --upload_rate UPLOAD_RATE
                        Select requested upload rate. Kbps, Mbps, Gbps units
                        supported. Default is 0Kbps
  --dut_model DUT_MODEL
                        AP name and Model
  --ssid_dut_2g SSID_DUT_2G
                        ssid name to be tested Ex. Netgear_2G
  --ssid_dut_5g SSID_DUT_5G
                        ssid name to be tested Ex. Netgear_5G
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir <where to pull reports to>
                        default '' put where dataplane script run from
  -s STATIONS, --stations STATIONS
                        If specified, these stations will be used. If not
                        specified, all available stations will be selected.
                        Example: 1.1.sta001,1.1.wlan0,...
  --help_summary        Show summary of what this script does
```

py-scripts/lf_wifi_capacity_test.py

Example report: wifi_capacity.pdf

```
usage: lf_wifi_capacity_test.py [-h] [-m MGR] [-o PORT] [--lf_user LF_USER]
                                [--lf_password LF_PASSWORD] [-i INSTANCE_NAME]
                                [-c CONFIG_NAME] [-r] [--load_old_cfg]
                                [--enable ENABLE] [--disable DISABLE]
                                [--set SET SET] [--raw_line RAW_LINE]
                                [--raw_lines_file RAW_LINES_FILE]
                                [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                                [-u UPSTREAM] [-b BATCH_SIZE] [-l LOOP_ITER]
                                [-p PROTOCOL] [-d DURATION]
                                [--verbosity VERBOSITY]
                                [--download_rate DOWNLOAD_RATE]
                                [--upload_rate UPLOAD_RATE] [--sort SORT]
                                [-s STATIONS] [-cs] [-radio RADIO]
                                [-ssid SSID] [-security SECURITY]
                                [-paswd PASWD] [--report_dir REPORT_DIR]
                                [--scenario SCENARIO]
                                [--graph_groups GRAPH_GROUPS]
                                [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                                [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                [--num_stations NUM_STATIONS]
                                [--start_id START_ID] [--log_level LOG_LEVEL]
                                [--help_summary]

NAME: lf_wifi_capacity_test.py

PURPOSE: This script runs wifi capacity test on the existing stations or runs on the stations specified
(if --stations argument is mentioned or stations can be created using -cs with stations names mentioned with --stations)
by creating layer3 cross connects and generates html and pdf report.

EXAMPLE:
example 1:
./lf_wifi_capacity_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge --instance_name wct_instance --co

example 2:
./lf_wifi_capacity_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge --instance_name wct_instance --co

example 3:
./lf_wifi_capacity_test.py --mgr 192.168.200.165 --upstream 1.1.eth1 --batch_size 1,5 --protocol UDP-IPv4 --duration 30000
 --upload_rate 1Gbps --download_rate 1Gbps --raw_line 'ip_tos: 128' --raw_line 'do_pf: 1' --raw_line 'pf_min_period_dl: 100'
  --raw_line 'pf_min_period_ul: 300' --raw_line 'pf_max_reconnects: 3' --num_stations 5 --start_id 333 --create_stations
  --radio wiphy0 --ssid Netgear-5g --security wpa2 --paswd sharedsecret --test_rig Testbed-01 --set DUT_NAME linksys-8450
  --pull_report

SCRIPT_CLASSIFICATION :  Test
SCRIPT_CATEGORIES:   Performance,  Functional,  KPI Generation,  Report Generation

NOTES: This script is used to automate wifi capacity tests.You need a configured upstream to run the script.
To Run this script gui should be opened with
192.168.200.147:1
    path: cd LANforgeGUI_5.4.3 (5.4.3 can be changed with GUI version)
          pwd (Output : /home/lanforge/LANforgeGUI_5.4.3)
          ./lfclient.bash -cli-socket 3990

This is a test file which will run a wifi capacity test.
    ex. on how to run this script (if stations are available in lanforge):

    ./lf_wifi_capacity_test.py --mgr localhost --port 8080 --lf_user lanforge --lf_password lanforge              --instance_name
    --pull_report == If specified, this will pull reports from lanforge to your code directory,
                     from where you are running this code

    --stations == Enter stations to use for wifi capacity

    --set DUT_NAME XXXX == Determines which DUT the wifi capacity test should use to get details on

STATUS: BETA RELEASE

VERIFIED_ON:
Working date - 11/05/2023
Build version - 5.4.6
kernel version - 6.2.14+

LICENSE:
    Free to distribute and modify. LANforge systems must be licensed.
    Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False



optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lf_user LF_USER     LANforge username to pull reports
  --lf_password LF_PASSWORD
```

```
                        LANforge Password to pull reports
  -i INSTANCE_NAME, --instance_name INSTANCE_NAME
                        create test instance
  -c CONFIG_NAME, --config_name CONFIG_NAME
                        Config file name
  -r, --pull_report     pull reports from lanforge (by default: False)
  --load_old_cfg        Should we first load defaults from previous run of the capacity test?  Default is False
  --enable ENABLE       Specify options to enable (set cfg-file value to 1).  See example raw text config for possible options.  M
  --disable DISABLE     Specify options to disable (set value to 0).  See example raw text config for possible options.  May be sp
  --set SET SET         Specify options to set values based on their label in the GUI. Example: --set 'Basic Client Connectivity'
  --raw_line RAW_LINE   Specify lines of the raw config file.  Example: --raw_line 'test_rig: Ferndale-01-Basic'  See example raw
  --raw_lines_file RAW_LINES_FILE
                        Specify a file of raw lines to apply.
  --test_rig TEST_RIG   Specify the test rig info for reporting purposes, for instance:  testbed-01
  --test_tag TEST_TAG   Specify the test tag info for reporting purposes, for instance:  testbed-01
  -u UPSTREAM, --upstream UPSTREAM
                        Upstream port for wifi capacity test ex. 1.1.eth1
  -b BATCH_SIZE, --batch_size BATCH_SIZE
                        station increment ex. 1,2,3
  -l LOOP_ITER, --loop_iter LOOP_ITER
                        Loop iteration ex. 1
  -p PROTOCOL, --protocol PROTOCOL
                        Protocol ex.TCP-IPv4
  -d DURATION, --duration DURATION
                        duration in ms. ex. 5000
  --verbosity VERBOSITY
                        Specify verbosity of the report values 1 - 11 default 5
  --download_rate DOWNLOAD_RATE
                        Select requested download rate.  Kbps, Mbps, Gbps units supported.  Default is 1Gbps
  --upload_rate UPLOAD_RATE
                        Select requested upload rate.  Kbps, Mbps, Gbps units supported.  Default is 10Mbps
  --sort SORT           Select station sorting behaviour:  none | interleave | linear  Default is interleave.
  -s STATIONS, --stations STATIONS
                        If specified, these stations will be used.  If not specified, all available stations will be selected.  Ex
  -cs, --create_stations
                        create stations in lanforge (by default: False)
  -radio RADIO, --radio RADIO
                        create stations in lanforge at this radio (by default: wiphy0)
  -ssid SSID, --ssid SSID
                        ssid name
  -security SECURITY, --security SECURITY
                        ssid Security type
  -paswd PASWD, --paswd PASWD, -passwd PASWD, --passwd PASWD
                        ssid Password
  --report_dir REPORT_DIR
  --scenario SCENARIO
  --graph_groups GRAPH_GROUPS
                        File to save graph groups to
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir <where to pull reports to>  default '' put where dataplane script run from
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --num_stations NUM_STATIONS
                        Specify the number of stations need to be create.
  --start_id START_ID   Specify the station starting id
                         e.g: --start_id <value> default 0
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --help_summary        Show summary of what this script does
```

py-scripts/measure_station_time_up.py

```
usage: measure_station_time_up.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                                  [-u UPSTREAM_PORT]
                                  [--num_stations NUM_STATIONS]
                                  [--test_id TEST_ID] [-d]
                                  [--log_level LOG_LEVEL]
                                  [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                  [--proxy [PROXY]]
                                  [--debugging DEBUGGING [DEBUGGING ...]]
                                  [--debug_log DEBUG_LOG] [--no_cleanup]
                                  [--help_summary] [--radio RADIO]
                                  [--security SECURITY] [--ssid SSID]
                                  [--passwd PASSWD]
                                  [--report_file REPORT_FILE]
                                  [--database DATABASE] [--radio2 RADIO2]

       measure_station_time_up.py
--------------------
Command example:
./measure_station_time_up.py
    --radio wiphy0
    --num_stations 3
    --security open
    --ssid netgear
    --passwd BLANK
    --debug
    --outfile


optional arguments:
  -h, --help            show this help message and exit
  --database DATABASE   Which database to load
  --radio2 RADIO2       second radio to create stations on

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
```

```
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

required arguments:
  --report_file REPORT_FILE
                        where you want to store results

        Measure how long it takes to up stations
```

py-scripts/modify_station.py

```
usage: modify_station.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                         [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                         [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                         [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                         [--proxy [PROXY]]
                         [--debugging DEBUGGING [DEBUGGING ...]]
                         [--debug_log DEBUG_LOG] [--no_cleanup]
                         [--help_summary] [--radio RADIO]
                         [--security SECURITY] [--ssid SSID] [--passwd PASSWD]
                         [--enable_flag ENABLE_FLAG]
                         [--disable_flag DISABLE_FLAG] [--station STATION]
                         [--set_state {DOWN,UP,down,up}] [--mac MAC]
                         [--mode MODE] [--bssid BSSID] [--ip IP]
                         [--netmask NETMASK] [--gateway GATEWAY]
                         [--channel CHANNEL] [--txpower TXPOWER]
                         [--antennas {All,1x1,2x2,3x3,4x4,8x8}]
                         [--country {US,AL,DZ,AR,BD,AM,AU,AT,AZ,BH,BB,BY,BE,BZ,BO,BA,BR,BN,BG,CA,CL,CN,CO,CR,HR,CY,CZ,DK,DO,EC,EG,
                         [--list_stations] [--list_ports]

       modify_station.py
       --------------------
       Command example:
       ./modify_station.py --mgr localhost --list_ports

       ./modify_station.py --mgr localhost --list_stations

       ./modify_station.py
           --radio         wiphy0
           --station       1.1.sta0000
           --set_state     up|down
           --security      open
           --ssid          netgear
           --passwd        BLANK
           --enable_flag   osen_enable
           --disable_flag  ht160_enable
           --bssid         00:11:22:33:44:55
           --mode          abgnAX
           --rate          [mcs rate available from txo-feautres]
           --ip            192.168.45.2
           --netmask       255.255.255.0
           --gateway       192.168.45.1
           --channel       6
           --txpower       24
           --antennas      2x2
           --country       [soon] US
           --debug
       --------------------
       Station flags are currently defined as:
       wpa_enable           | 0x10         # Enable WPA
       custom_conf          | 0x20         # Use Custom wpa_supplicant config file.
       wep_enable           | 0x200        # Use wpa_supplicant configured for WEP encryption.
       wpa2_enable          | 0x400        # Use wpa_supplicant configured for WPA2 encryption.
       ht40_disable         | 0x800        # Disable HT-40 even if hardware and AP support it.
       scan_ssid            | 0x1000       # Enable SCAN-SSID flag in wpa_supplicant.
       passive_scan         | 0x2000       # Use passive scanning (don't send probe requests).
       disable_sgi          | 0x4000       # Disable SGI (Short Guard Interval).
       lf_sta_migrate       | 0x8000       # OK-To-Migrate (Allow station migration between LANforge radios)
       verbose              | 0x10000      # Verbose-Debug:  Increase debug info in wpa-supplicant and hostapd logs.
       80211u_enable        | 0x20000      # Enable 802.11u (Interworking) feature.
       80211u_auto          | 0x40000      # Enable 802.11u (Interworking) Auto-internetworking feature.  Always enabled currentl
       80211u_gw            | 0x80000      # AP Provides access to internet (802.11u Interworking)
       80211u_additional    | 0x100000     # AP requires additional step for access (802.11u Interworking)
       80211u_e911          | 0x200000     # AP claims emergency services reachable (802.11u Interworking)
       80211u_e911_unauth   | 0x400000     # AP provides Unauthenticated emergency services (802.11u Interworking)
       hs20_enable          | 0x800000     # Enable Hotspot 2.0 (HS20) feature.  Requires WPA-2.
       disable_gdaf         | 0x1000000    # AP:  Disable DGAF (used by HotSpot 2.0).
       8021x_radius         | 0x2000000    # Use 802.1x (RADIUS for AP).
       80211r_pmska_cache   | 0x4000000    # Enable opportunistic PMSKA caching for WPA2 (Related to 802.11r).
       disable_ht80         | 0x8000000    # Disable HT80 (for AC chipset NICs only)
       ibss_mode            | 0x20000000   # Station should be in IBSS mode.
       osen_enable          | 0x40000000   # Enable OSEN protocol (OSU Server-only Authentication)
       disable_roam         | 0x80000000   # Disable automatic station roaming based on scan results.
       ht160_enable         | 0x100000000  # Enable HT160 mode.
```

```
       disable_fast_reauth  | 0x200000000  # Disable fast_reauth option for virtual stations.
       mesh_mode            | 0x400000000  # Station should be in MESH mode.
       power_save_enable    | 0x800000000  # Station should enable power-save.  May not work in all drivers/configurations.
       create_admin_down    | 0x1000000000 # Station should be created admin-down.
       wds-mode             | 0x2000000000 # WDS station (sort of like a lame mesh), not supported on ath10k
       no-supp-op-class-ie  | 0x4000000000 # Do not include supported-oper-class-IE in assoc requests.  May work around AP bugs.
       txo-enable           | 0x8000000000 # Enable/disable tx-offloads, typically managed by set_wifi_txo command
       use-wpa3             | 0x10000000000 # Enable WPA-3 (SAE Personal) mode.
       use-bss-transition   | 0x80000000000 # Enable BSS transition.
       disable-twt          | 0x100000000000 # Disable TWT mode

optional arguments:
  -h, --help            show this help message and exit
  --country {US,AL,DZ,AR,BD,AM,AU,AT,AZ,BH,BB,BY,BE,BZ,BO,BA,BR,BN,BG,CA,CL,CN,CO,CR,HR,CY,CZ,DK,DO,EC,EG,SV,EE,FI,FR,GE,DE,GR,GT,
                        Set the country code for the lanforge resource. This needs to set all radios on the resource to the same c

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

optional arguments:
  --enable_flag ENABLE_FLAG
                        station flags to add
  --disable_flag DISABLE_FLAG
                        station flags to disable
  --station STATION     station to modify
  --set_state {DOWN,UP,down,up}, --state {DOWN,UP,down,up}, --up {DOWN,UP,down,up}
                        admin port UP or DOWN
  --mac MAC             MAC address of the station
  --mode MODE           set station wifi mode: AUTO, 802.11a, b, g, abg, abgn, bgn, bg, abgnAC, anAC, an, bgnAC, abgnAX, bgnAX, an
  --bssid BSSID         specify the BSSID of the AP to associate with, or DEFAULT
  --ip IP               specify IP to apply to port (ipv4 1.2.3.4 or DHCP)
  --netmask NETMASK     specify netmask to apply to port
  --gateway GATEWAY     specify gateway to apply to port
  --channel CHANNEL     specify channel for radio, requires --radio
  --txpower TXPOWER     specify txpower for radio, requires --radio, use 0-25 or DEFAULT or -1
  --antennas {All,1x1,2x2,3x3,4x4,8x8}, --antenna {All,1x1,2x2,3x3,4x4,8x8}
                        specify antenna diversity for radio (NSS), requires --radio
  --list_stations       lists station by Eid
  --list_ports          lists ports by Eid

        Modify stations on a system. Use the enable_flag to create a flag on a station. Turn off a flag with          the disable
```

py-scripts/modify_vap.py

```
usage: modify_vap.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [-u UPSTREAM_PORT]
                     [--num_stations NUM_STATIONS] [--test_id TEST_ID] [-d]
                     [--log_level LOG_LEVEL]
                     [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                     [--proxy [PROXY]] [--debugging DEBUGGING [DEBUGGING ...]]
                     [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
                     [--radio RADIO] [--security SECURITY] [--ssid SSID]
                     [--passwd PASSWD] [--enable_flag ENABLE_FLAG]
                     [--disable_flag DISABLE_FLAG] [--vap VAP] [--mac MAC]

       modify_vap.py
       --------------------
       Command example:
       ./modify_vap.py
           --radio wiphy0
           --vap 1.1.vap0000
           --security open
           --ssid netgear
           --passwd BLANK
           --enable_flag osen_enable
           --disable_flag ht160_enable
           --debug
       --------------------
       AP flags are currently defined as:

enable_wpa           | 0x10            # Enable WPA
hostapd_config       | 0x20            # Use Custom hostapd config file.
enable_80211d        | 0x40            # Enable 802.11D to broadcast country-code & channels in VAPs
```

```
short_preamble       | 0x80            # Allow short-preamble
pri_sec_ch_enable    | 0x100           # Enable Primary/Secondary channel switch.
wep_enable           | 0x200           # Enable WEP Encryption
wpa2_enable          | 0x400           # Enable WPA2 Encryption
disable_ht40         | 0x800           # Disable HT-40 (will use HT-20 if available).
verbose              | 0x10000         # Verbose-Debug:  Increase debug info in wpa-supplicant and hostapd logs.
80211u_enable        | 0x20000         # Enable 802.11u (Interworking) feature.
80211u_auto          | 0x40000         # Enable 802.11u (Interworking) Auto-internetworking feature.  Always enabled currently.
80211u_gw            | 0x80000         # AP Provides access to internet (802.11u Interworking)
80211u_additional    | 0x100000        # AP requires additional step for access (802.11u Interworking)
80211u_e911          | 0x200000        # AP claims emergency services reachable (802.11u Interworking)
80211u_e911_unauth   | 0x400000        # AP provides Unauthenticated emergency services (802.11u Interworking)
hs20_enable          | 0x800000        # Enable Hotspot 2.0 (HS20) feature.  Requires WPA-2.
disable_dgaf         | 0x1000000       # AP Disable DGAF (used by HotSpot 2.0).
8021x_radius         | 0x2000000       # Use 802.1x (RADIUS for AP).
80211r_pmska_cache   | 0x4000000       # Enable oportunistic PMSKA caching for WPA2 (Related to 802.11r).
disable_ht80         | 0x8000000       # Disable HT80 (for AC chipset NICs only)
80211h_enable        | 0x10000000      # Enable 802.11h (needed for running on DFS channels)  Requires 802.11d.
osen_enable          | 0x40000000      # Enable OSEN protocol (OSU Server-only Authentication)
ht160_enable         | 0x100000000     # Enable HT160 mode.
create_admin_down    | 0x1000000000    # Station should be created admin-down.
use-wpa3             | 0x10000000000   # Enable WPA-3 (SAE Personal) mode.
use-bss-load         | 0x20000000000   # Enable BSS Load IE in Beacons and Probe Responses (.11e).
use-rrm-report       | 0x40000000000   # Enable Radio measurements IE in beacon and probe responses.
use-bss-transition   | 0x80000000000   # Enable BSS transition.



optional arguments:
  -h, --help            show this help message and exit

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

optional arguments:
  --enable_flag ENABLE_FLAG
                        VAP flags to add
  --disable_flag DISABLE_FLAG
                        VAP flags to disable
  --vap VAP             VAP to modify
  --mac MAC

        Modify VAPs on a system. Use the enable_flag to create a flag on a VAP. Turn off a flag with          the disable_flag op
```

py-scripts/monitor_cx.py

```
usage: ./monitor_cx.py [-h] [--host HOST] [--cx_names CX_NAMES [CX_NAMES ...]]
                       [--csv_file CSV_FILE] [--quit QUIT] [--debug]
                       [--log_level LOG_LEVEL] [--help_summary]

monitors connections and prints data to a csv file

optional arguments:
  -h, --help            show this help message and exit
  --host HOST, --mgr HOST
                        specify the GUI to connect to, assumes port 8080
  --cx_names CX_NAMES [CX_NAMES ...]
                        spsace or comma separated list of cx names, or ALL
  --csv_file CSV_FILE   csv filename to save data to
  --quit QUIT           when to exit the script: all_cx_stopped: when all connections stop
  --debug               turn on debugging
  --log_level LOG_LEVEL
  --help_summary        Show summary of what this script does
```

py-scripts/raw_cli.py

```
usage: ./raw_cli.py [-h] [--host HOST] [--help_summary] [--raw RAW]
                    [--cmd CMD] [--arg ARG [ARG ...]] [--debug]

tests creating raw command

optional arguments:
```

```
  -h, --help            show this help message and exit
  --host HOST, --mgr HOST
                        specify the GUI to connect to, assumes port 8080
  --help_summary        purpose of the script
  --raw RAW             full CLI command to execute, including all arguments
  --cmd CMD             CLI command, where arguments to the command are provided using --arg parameters
  --arg ARG [ARG ...], --param ARG [ARG ...]
                        paramets with value, eg: --arg "alias bartleby" --arg "max-txbps 1000000"
  --debug, -d           turn on debugging
```

py-scripts/run_cv_scenario.py

```
usage: run_cv_scenario.py [-h] [--lfmgr LFMGR] [--port PORT]
                          [--lanforge_db LANFORGE_DB]
                          [--cv_scenario CV_SCENARIO] [--cv_test CV_TEST]
                          [--test_profile TEST_PROFILE]
                          [--report_verbosity REPORT_VERBOSITY]
                          [--leave_test_open] [--toggle_auto_save]
                          [--click_save] [--pre PRE [PRE ...]]
                          [--add ADD [ADD ...]] [--post POST [POST ...]]
                          [--debug] [--log_level LOG_LEVEL]
                          [--report_dir REPORT_DIR]
                          [--report_file_name REPORT_FILE_NAME]
                          [--help_summary]

Chamber View testing script:  Load a Chamber View (CV) scenario, build it, and run a test.

Example of loading a CV scenario, build it, and run the  WiFi Capacity
test using a previously saved profile:
    ./run_cv_scenario.py --lfmgr 127.0.0.1 --lanforge_db 'handsets' \
        --cv_scenario Mobile20 \
        --cv_test 'WiFi Capacity' \
        --test_profile 'test-20'

Example of running a WiFi Capacity test using the current CV scenario.
This combination of parameters will not re-generate the present CV scenario:
    ./run_cv_scenario.py --lanforge_db DFLT \
        --cv_scenario CURRENT \
        --cv_test 'WiFi Capacity' \
        --test_profile DEFAULT

optional arguments:
  -h, --help            show this help message and exit
  --lfmgr LFMGR, -m LFMGR
                        address of the LANforge GUI machine (localhost is default)
  --port PORT, -o PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lanforge_db LANFORGE_DB, -d LANFORGE_DB
                        Saved Test Configuration (database)
                        Use DFLT for present configuration.
                        See Status tab->Saved Test Configurations for saved configs.
  --cv_scenario CV_SCENARIO, -c CV_SCENARIO
                        Name of CV Test Scenario. See CV->Manage Scenarios for scenario names.
                        Use DFLT to avoid loading a new scenario (default behavior).
  --cv_test CV_TEST, -n CV_TEST
                        Chamber View test name, or QUIT to run no tests.
                        This is required.
  --test_profile TEST_PROFILE, -s TEST_PROFILE
                        Name of the saved CV test profile. Default is 'DEFAULT'
  --report_verbosity REPORT_VERBOSITY, --verbosity REPORT_VERBOSITY, -v REPORT_VERBOSITY
                        Report verbosity level: 0 - 11
  --leave_test_open, --leave_open, --leave
                        leave test window open when test completes
  --toggle_auto_save    This toggles the Auto Save Report checkbox. If your test profile was saved with Auto Save Report selected,
  --click_save          Show the save-report dialog as if you clicked 'Save HTML'. Both the HTML and PDF reports are created, the
  --pre PRE [PRE ...], -x PRE [PRE ...]
                        One or more GUI-CLI arguments before starting the test, after test profile is loaded.Multiple commands all
  --add ADD [ADD ...], -y ADD [ADD ...]
                        One or more GUI-CLI arguments immediately as the test starts, before the report ends.Multiple commands all
  --post POST [POST ...], -z POST [POST ...]
                        One or more GUI-CLI arguments after the test finishes, before the test window is closed or click_save. Mul
  --debug               Enable debugging
  --log_level LOG_LEVEL
                        debug message verbosity
  --report_dir REPORT_DIR, --dir REPORT_DIR
                        report directory
  --report_file_name REPORT_FILE_NAME, --report_file REPORT_FILE_NAME
                        name of the report file
  --help_summary        Show summary of what this script does
```

py-scripts/run_voip_cx.py

```
usage: ./run_voip_cx.py [-h] [--host HOST] [--csv_file CSV_FILE]
                        [--cx_list CX_NAMES_STR] [--debug]
                        [--log_level LOG_LEVEL]
                        [--num_calls [NUM_CALLS [NUM_CALLS ...]]]
                        [--side_a_phone_nums [SIDE_A_PHONE_NUMS [SIDE_A_PHONE_NUMS ...]]]
                        [--side_b_phone_nums [SIDE_B_PHONE_NUMS [SIDE_B_PHONE_NUMS ...]]]
                        [--side_a_mobile_bt_macs [SIDE_A_MOBILE_BT_MACS [SIDE_A_MOBILE_BT_MACS ...]]]
                        [--side_b_mobile_bt_macs [SIDE_B_MOBILE_BT_MACS [SIDE_B_MOBILE_BT_MACS ...]]]
                        [--help_summary]

optional arguments:
  -h, --help            show this help message and exit
  --host HOST, --mgr HOST
                        URL of the LANforge GUI machine (localhost is default, http://localhost:8080)
  --csv_file CSV_FILE   name of the csv output file
  --cx_list CX_NAMES_STR, --cx_names CX_NAMES_STR
                        comma separated list of voip connection names, or 'ALL'
  --debug               Enable debugging
  --log_level LOG_LEVEL
                        debug message verbosity
  --num_calls [NUM_CALLS [NUM_CALLS ...]], --loop_call_count [NUM_CALLS [NUM_CALLS ...]]
                        Number of calls to make in looped mode. If one number specified, applies to all CXs specified. If more tha
  --side_a_phone_nums [SIDE_A_PHONE_NUMS [SIDE_A_PHONE_NUMS ...]], --side_a_phone_numbers [SIDE_A_PHONE_NUMS [SIDE_A_PHONE_NUMS ..
                        List of phone numbers to configure on side A VoIP endpoints. Order and length must match the order of conn
  --side_b_phone_nums [SIDE_B_PHONE_NUMS [SIDE_B_PHONE_NUMS ...]], --side_b_phone_numbers [SIDE_B_PHONE_NUMS [SIDE_B_PHONE_NUMS ..
                        List of phone numbers to configure on side A VoIP endpoints. Order and length must match the order of conn
```

```
  --side_a_mobile_bt_macs [SIDE_A_MOBILE_BT_MACS [SIDE_A_MOBILE_BT_MACS ...]]
                        List of Bluetooth MAC addresses to configure on side A VoIP endpoints. Order and length must match the ord
  --side_b_mobile_bt_macs [SIDE_B_MOBILE_BT_MACS [SIDE_B_MOBILE_BT_MACS ...]]
                        List of Bluetooth MAC addresses to configure on side A VoIP endpoints. Order and length must match the ord
  --help_summary        Show summary of what this script does
```

py-scripts/rvr_scenario.py

```
usage: rvr_scenario.py [-h] [-m LFMGR] [-o PORT] [--lanforge_db LANFORGE_DB]
                       [-c CV_SCENARIO] [-n CV_TEST] [-s TEST_PROFILE]
                       [--help_summary]

LANforge Reporting Script:  Load a scenario and run a RvR report
Example:
./rvr_scenario.py --lfmgr 127.0.0.1 --lanforge_db 'handsets' --cv_test  --test_scenario 'test-20'

optional arguments:
  -h, --help            show this help message and exit
  -m LFMGR, --lfmgr LFMGR
                        address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  --lanforge_db LANFORGE_DB, --db LANFORGE_DB, --lanforge_scenario LANFORGE_DB
                        Name of test scenario database (see Status Tab)
  -c CV_SCENARIO, --cv_scenario CV_SCENARIO
                        Name of Chamber View test scenario (see CV Manage Scenarios)
  -n CV_TEST, --cv_test CV_TEST
                        Chamber View test
  -s TEST_PROFILE, --test_profile TEST_PROFILE, --test_settings TEST_PROFILE
                        Name of the saved CV test settings
  --help_summary        Show summary of what this script does
```

py-scripts/scenario.py

```
usage: scenario.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [--debug]
                   [--log_level LOG_LEVEL]
                   [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                   [--proxy [PROXY]] [--help_summary] [--load LOAD]
                   [--action ACTION] [--clean_dut] [--clean_chambers]
                   [--start START | --quiesce QUIESCE | --stop STOP]
                   [--timeout TIMEOUT] [--quit_on_phantom]
                   [--check_phantom CHECK_PHANTOM [CHECK_PHANTOM ...]]

scenario.py
    --------------------
    Generic command example:
    scenario.py --load db1 --action overwrite --clean_dut --clean_chambers

    scenario.py --start test_group1

    scenario.py --quiesce test_group1

    scenario.py --stop test_group1


optional arguments:
  -h, --help            show this help message and exit
  --load LOAD           name of database to load
  --action ACTION       action to take with database {overwrite | append}
  --clean_dut           use to cleanup DUT will be when overwrite is selected, otherwise they will be kept
  --clean_chambers      use to cleanup Chambers will be when overwrite is selected, otherwise they will be kept
  --start START         name of test group to start
  --quiesce QUIESCE     name of test group to quiesce
  --stop STOP           name of test group to stop
  --timeout TIMEOUT     Stop trying to load scenario after this many seconds
  --quit_on_phantom     do not load the database if a phantom port is present
  --check_phantom CHECK_PHANTOM [CHECK_PHANTOM ...]
                        check if these ports are phantom

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_bare_argparse found in /lanforge-scripts/py-json/LANfo
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --debug, -d           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80 or https://user:pass@proxy.localnet:3128
  --help_summary        Show summary of what this script does

Load a database file and control test groups
```

py-scripts/ssh_remote.py

```
usage: ssh_remote.py [-h] [--prog PROG] [--remote_args REMOTE_ARGS] [--ip IP]
                     [--username USERNAME] [--password PASSWORD]
                     [--help_summary]

 Run command against remoate machine over ssh


optional arguments:
  -h, --help            show this help message and exit
  --prog PROG           Remote command to execute
  --remote_args REMOTE_ARGS
                        Arguments for remote command
  --ip IP               IP address of remote system
  --username USERNAME   User-name for remote machine
  --password PASSWORD   Password for remote machine
  --help_summary        Show summary of what this script does

 Run command against remoate machine over ssh
```

py-scripts/sta_connect2.py

```
usage: sta_connect2.py [-h] [-m MGR] [-o PORT] [-u USER] [-p PASSWD]
                       [--resource RESOURCE]
```

```
                      [--upstream_resource UPSTREAM_RESOURCE]
                      [--upstream_port UPSTREAM_PORT] [--radio RADIO]
                      [--sta_mode STA_MODE] [--dut_ssid DUT_SSID]
                      [--dut_security DUT_SECURITY] [--dut_passwd DUT_PASSWD]
                      [--dut_bssid DUT_BSSID] [--download_bps DOWNLOAD_BPS]
                      [--upload_bps UPLOAD_BPS] [--side_a_pdu SIDE_A_PDU]
                      [--side_b_pdu SIDE_B_PDU] [--runtime_sec RUNTIME_SEC]
                      [--debug] [--prefix PREFIX]
                      [--bringup_time BRINGUP_TIME]
                      [--monitor_interval MONITOR_INTERVAL]
                      [--debug_log DEBUG_LOG] [--no_cleanup]
                      [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                      [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                      [--dut_hw_version DUT_HW_VERSION]
                      [--dut_sw_version DUT_SW_VERSION]
                      [--dut_model_num DUT_MODEL_NUM]
                      [--dut_serial_num DUT_SERIAL_NUM]
                      [--test_priority TEST_PRIORITY] [--test_id TEST_ID]
                      [--csv_outfile CSV_OUTFILE] [--log_level LOG_LEVEL]
                      [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                      [--help_summary]

---------------------------
LANforge Unit Test:  Connect Station to AP - sta_connect2.py
---------------------------
Summary:
This will create a station, create TCP and UDP traffic, run it a short amount of time, and verify whether traffic
was sent and received.  It also verifies the station connected to the requested BSSID if bssid is specified
as an argument. The script will clean up the station and connections at the end of the test.
---------------------------
CLI Example:
./sta_connect2.py --mgr localhost --dut_ssid <ssid> --dut_passwd <passwd> --dut_security wpa2
--upstream_port eth1 --radio wiphy1

CLI Example for kpi.csv report output:
./sta_connect2.py --mgr localhost --dut_ssid <ssid> --dut_passwd <passwd> --dut_security wpa2
--upstream_port eth2  --csv_outfile sta_connect2.csv --test_rig LF-Lab --test_tag L3 --dut_hw_version Linux
--dut_model_num 1 --dut_sw_version 5.4.5 --dut_serial_num 1234

CLI Example for kpi.csv, variable tx/rx rates, and pdu size:
./sta_connect2.py --mgr localhost --dut_ssid <ssid> --dut_passwd <passwd> --dut_security wpa2
--upstream_port eth2  --download_bps 768000 --upload_bps 256000 --side_a_pdu 300 --side_b_pdu 750
--csv_outfile sta_connect2.csv --test_rig LF-Lab --test_tag L3 --dut_hw_version Linux --dut_model_num 1
--dut_sw_version 5.4.5 --dut_serial_num 1234

Note: --sta_mode use values in second column

        AUTO        |  0        #  802.11g
        802.11a     |  1        #  802.11a
        b           |  2        #  802.11b
        g           |  3        #  802.11g
        abg         |  4        #  802.11abg
        abgn        |  5        #  802.11abgn
        bgn         |  6        #  802.11bgn
        bg          |  7        #  802.11bg
        abgnAC      |  8        #  802.11abgn-AC
        anAC        |  9        #  802.11an-AC
        an          | 10        #  802.11an
        bgnAC       | 11        #  802.11bgn-AC
        abgnAX      | 12        #  802.11abgn-AX
                                #     a/b/g/n/AC/AX (dual-band AX) support
        bgnAX       | 13        #  802.11bgn-AX
        anAX        | 14        #  802.11an-AX
        aAX         | 15        #  802.11a-AX (6E disables /n and /ac)

---------------------------

optional arguments:
  -h, --help            show this help message and exit
  -m MGR, --mgr MGR     address of the LANforge GUI machine (localhost is default)
  -o PORT, --port PORT  IP Port the LANforge GUI is listening on (8080 is default)
  -u USER, --user USER  TBD: credential login/username
  -p PASSWD, --passwd PASSWD
                        TBD: credential password
  --resource RESOURCE   LANforge Station resource ID to use, default is 1
  --upstream_resource UPSTREAM_RESOURCE
                        LANforge Ethernet port resource ID to use, default is 1
  --upstream_port UPSTREAM_PORT
                        LANforge Ethernet port name, default is eth2
  --radio RADIO         LANforge radio to use, default is wiphy0
  --sta_mode STA_MODE   LANforge station-mode setting (see add_sta LANforge CLI documentation, default is 0 (auto))
  --dut_ssid DUT_SSID   DUT SSID
  --dut_security DUT_SECURITY
                        DUT security: open, wpa, wpa2, wpa3, owe
  --dut_passwd DUT_PASSWD
                        DUT PSK password.  Do not set for OPEN auth
  --dut_bssid DUT_BSSID
                        DUT BSSID to which we expect to connect.
  --download_bps DOWNLOAD_BPS
                        Set the minimum bps value on test endpoint A. Default: 25g000
  --upload_bps UPLOAD_BPS
                        Set the minimum bps value on test endpoint B. Default: 256000
  --side_a_pdu SIDE_A_PDU
                        Set the minimum pdu value on test endpoint A. Default: 1200
  --side_b_pdu SIDE_B_PDU
                        Set the minimum pdu value on test endpoint B. Default: 1500
  --runtime_sec RUNTIME_SEC
                        Set test duration time. Default: 60 seconds
  --debug               enable debugging
  --prefix PREFIX       Station prefix. Default: 'sta'
  --bringup_time BRINGUP_TIME
                        Seconds to wait for stations to associate and aquire IP. Default: 300
  --monitor_interval MONITOR_INTERVAL
                        How frequently you want to append to your database
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
```

```
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path, primary use when running test in test suite
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differenciate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number / serial number of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --test_id TEST_ID     test-id for kpi.csv,  script or test name
  --csv_outfile CSV_OUTFILE
                        --csv_outfile <prepend input to generated file for csv data>
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --help_summary        Show summary of what this script does

---------------------------
CLI Example:
./sta_connect2.py --mgr localhost --dut_ssid <ssid> --dut_passwd <passwd> --dut_security wpa2
--upstream_port eth2 --radio wiphy1
```

py-scripts/sta_connect_bssid_mac.py

```
usage: sta_connect_bssid_mac.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                                [--debug] [--log_level LOG_LEVEL]
                                [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                [--proxy [PROXY]] [--help_summary]
                                [--ssid SSID] [--passwd PASSWD]
                                [--security SECURITY] [--radio RADIO]
                                [--sta_name STA_NAME] [--bssid BSSID]
                                [--mac MAC] [-bt]

                                          --mgr localhost --mgr_port 8080
                                          --ssid "TestAP-Jitendra"
                                          --radio wiphy0
                                          --security "open" // "wpa" //  "wpa2"
                                          --passwd "BLANK"
                                          --bssid 78:d2:94:4f:20:c5,78:d2:94:4f:20:c5
                                          --sta_name "sta001,sta002"
                                          --mac 04:f0:21:89:3e:ea,04:f0:21:89:4e:ea
                                          --bss_trans


optional arguments:
  -h, --help            show this help message and exit
  --ssid SSID           --ssid
  --passwd PASSWD       --passwd
  --security SECURITY   --security
  --radio RADIO         --radio to use
  --sta_name STA_NAME   --num_client is number of stations
  --bssid BSSID         DUT BSSID to which we expect to connect
  --mac MAC             --mac to stations
  -bt, --bss_trans      To enable BSS transition.(by default: False)

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_bare_argparse found in /lanforge-scripts/py-json/LANfo
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --debug, -d           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80 or https://user:pass@proxy.localnet:3128
  --help_summary        Show summary of what this script does
```

py-scripts/sta_connect_example.py

```
usage: sta_connect_example.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                              [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                              [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                              [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                              [--proxy [PROXY]]
                              [--debugging DEBUGGING [DEBUGGING ...]]
                              [--debug_log DEBUG_LOG] [--no_cleanup]
                              [--help_summary] [--radio RADIO]
                              [--security SECURITY] [--ssid SSID]
                              [--passwd PASSWD]
                              [--test_duration TEST_DURATION]

optional arguments:
  -h, --help            show this help message and exit
  --test_duration TEST_DURATION
                        --test_duration sets the duration of the test

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
```

```
                        or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key
```

py-scripts/sta_connect_multi_example.py

```
usage: sta_connected_multip_example.py [-h] [--help_summary]

Example of how to instantiate StaConnect and run the test


optional arguments:
  -h, --help      show this help message and exit
  --help_summary  Show summary of what this script does

        sta_connected_multip_example.py
```

py-scripts/sta_scan_test.py

```
usage: sta_scan_test.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                        [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                        [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                        [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                        [--proxy [PROXY]]
                        [--debugging DEBUGGING [DEBUGGING ...]]
                        [--debug_log DEBUG_LOG] [--no_cleanup]
                        [--help_summary] [--radio RADIO] [--security SECURITY]
                        [--ssid SSID] [--passwd PASSWD]
                        [--use_existing_station] [--mode MODE]
                        [--sta_name STA_NAME [STA_NAME ...]]
                        [--csv_output CSV_OUTPUT] [--scan_time SCAN_TIME]

        Optionally creates a station with specified ssid info (can be real or fake ssid, if fake use open for security).
        If not creating a station, it can use existing station.
        Then starts a scan and waits 15 seconds, finally scan results are printed to console.

        Example:
        ./sta_scan_test.py --ssid test_name --security open --radio wiphy0
        ./sta_scan_test.py --sta_name 1.14.wlan0 1.1.wlan0 --use_existing_station --scan_time 5


optional arguments:
  -h, --help            show this help message and exit
  --use_existing_station
                        Use existing station instead of trying to create stations.
  --mode MODE           Used to force mode of stations
  --sta_name STA_NAME [STA_NAME ...]
                        Optional: User defined station names: 1.2.wlan0 1.3.wlan0
  --csv_output CSV_OUTPUT
                        Specify file to which csv output will be saved, otherwise print it in the terminal
  --scan_time SCAN_TIME
                        Specify time in seconds to wait for scan to complete.  Default is 15

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
```

```
                        WiFi passphrase/password/key

       Used to scan for ssids after creating a station
```

py-scripts/stations_connected.py

```
usage: stations_connected.py [-h] [--help_summary]

    Contains examples of using realm to query stations and get specific information from them


optional arguments:
  -h, --help      show this help message and exit
  --help_summary  Show summary of what this script does

       stations_connected.py
```

py-scripts/test_client_admission.py

```
usage: test_client_admission.py [-h] [-hst HOST] [-s SSID] [-pwd PASSWD]
                                [-sec SECURITY] [-rad RADIO]
                                [-num_sta NUM_STA] [--help_summary]

Client Admission Test Script

optional arguments:
  -h, --help            show this help message and exit
  -hst HOST, --host HOST
                        host name
  -s SSID, --ssid SSID  ssid for client
  -pwd PASSWD, --passwd PASSWD
                        password to connect to ssid
  -sec SECURITY, --security SECURITY
                        security
  -rad RADIO, --radio RADIO
                        radio at which client will be connected
  -num_sta NUM_STA, --num_sta NUM_STA
                        provide number of stations you want to create
  --help_summary        Show summary of what this script does
```

py-scripts/test_fileio.py

```
usage: test_fileio.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [--debug]
                      [--log_level LOG_LEVEL]
                      [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                      [--proxy [PROXY]] [--help_summary]
                      [--num_stations NUM_STATIONS] [--radio RADIO]
                      [--ssid SSID] [--passwd PASSWD] [--security SECURITY]
                      [-u UPSTREAM_PORT] [--test_duration TEST_DURATION]
                      [--fs_type FS_TYPE] [--min_rw_size MIN_RW_SIZE]
                      [--max_rw_size MAX_RW_SIZE]
                      [--min_file_size MIN_FILE_SIZE]
                      [--max_file_size MAX_FILE_SIZE]
                      [--min_read_rate_bps MIN_READ_RATE_BPS]
                      [--max_read_rate_bps MAX_READ_RATE_BPS]
                      [--min_write_rate_bps MIN_WRITE_RATE_BPS]
                      [--max_write_rate_bps MAX_WRITE_RATE_BPS]
                      [--directory DIRECTORY] [--server_mount SERVER_MOUNT]
                      [--macvlan_parent MACVLAN_PARENT]
                      [--first_port FIRST_PORT] [--num_ports NUM_PORTS]
                      [--connections_per_port CONNECTIONS_PER_PORT]
                      [--use_ports USE_PORTS] [--use_macvlans]
                      [--first_mvlan_ip FIRST_MVLAN_IP] [--netmask NETMASK]
                      [--gateway GATEWAY] [--use_test_groups]
                      [--read_only_test_group READ_ONLY_TEST_GROUP]
                      [--write_only_test_group WRITE_ONLY_TEST_GROUP]
                      [--mode MODE] [--test_rig TEST_RIG]
                      [--test_tag TEST_TAG] [--dut_hw_version DUT_HW_VERSION]
                      [--dut_sw_version DUT_SW_VERSION]
                      [--dut_model_num DUT_MODEL_NUM]
                      [--dut_serial_num DUT_SERIAL_NUM]
                      [--test_priority TEST_PRIORITY]
                      [--csv_outfile CSV_OUTFILE] [--test_id TEST_ID]
                      [--add_to_group ADD_TO_GROUP | --del_from_group DEL_FROM_GROUP]
                      [--cxs CXS]

test_fileio.py:
--------------------
Generic command layout:
./test_fileio.py --macvlan_parent <port> --num_ports <num ports> --use_macvlans
                 --first_mvlan_ip <first ip in series> --netmask <netmask to use> --gateway <gateway ip addr>

./test_fileio.py --macvlan_parent eth2 --num_ports 3 --use_macvlans --first_mvlan_ip 192.168.92.13
                 --netmask 255.255.255.0 --gateway 192.168.92.1

./test_fileio.py --radio 1.wiphy0 --test_duration 1m --macvlan_parent eth1 --num_ports 3 --use_macvlans
                 --use_ports eth1#0,eth1#1,eth1#2 --connections_per_port 2 --mode write

./test_fileio.py --radio 1.wiphy0 --test_duration 1m --macvlan_parent eth1 --num_ports 3 --use_macvlans
                 --first_mvlan_ip 10.40.3.100 --netmask 255.255.240.0 --gateway 10.40.0.1
                 --use_test_groups --write_only_test_group test_wo --read_only_test_group test_ro
                 --add_to_group test_wo --cxs test_wo0000,test_wo0001,test_wo0002

./test_fileio.py --radio 1.wiphy0 --test_duration 1m --macvlan_parent eth1 --num_ports 3 --use_macvlans
                 --use_ports eth1#0=10.40.3.103,eth1#1,eth1#2 --connections_per_port 2
                 --netmask 255.255.240.0 --gateway 10.40.0.1

optional arguments:
  -h, --help            show this help message and exit
  --num_stations NUM_STATIONS
                        Number of stations to create
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --ssid SSID           SSID for stations to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
```

```
                        WiFi passphrase/password/key
  --security SECURITY   security type to use for ssid { wep | wpa | wpa2 | wpa3 | open }
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --test_duration TEST_DURATION
                        sets the duration of the test
  --fs_type FS_TYPE     endpoint type
  --min_rw_size MIN_RW_SIZE
                        minimum read/write size
  --max_rw_size MAX_RW_SIZE
                        maximum read/write size
  --min_file_size MIN_FILE_SIZE
                        minimum file size
  --max_file_size MAX_FILE_SIZE
                        maximum file size
  --min_read_rate_bps MIN_READ_RATE_BPS
                        minimum bps read rate
  --max_read_rate_bps MAX_READ_RATE_BPS
                        maximum bps read rate
  --min_write_rate_bps MIN_WRITE_RATE_BPS
                        minimum bps write rate
  --max_write_rate_bps MAX_WRITE_RATE_BPS
                        maximum bps write rate
  --directory DIRECTORY
                        --directory directory to read/write in. Absolute path suggested
  --server_mount SERVER_MOUNT
                        --server_mount The server to mount, ex: 192.168.100.5/exports/test1
  --macvlan_parent MACVLAN_PARENT
                        specifies parent port for macvlan creation
  --first_port FIRST_PORT
                        specifies name of first port to be used
  --num_ports NUM_PORTS
                        number of ports to create
  --connections_per_port CONNECTIONS_PER_PORT
                        specifies number of connections to be used per port
  --use_ports USE_PORTS
                        list of comma separated ports to use with ips, '=' separates name and ip{ port_name1=ip_addr1,port_name1=i
  --use_macvlans        will create macvlans
  --first_mvlan_ip FIRST_MVLAN_IP
                        specifies first static ip address to be used or dhcp
  --netmask NETMASK     specifies netmask to be used with static ip addresses
  --gateway GATEWAY     specifies default gateway to be used with static addressing
  --use_test_groups     will use test groups to start/stop instead of single endps/cxs
  --read_only_test_group READ_ONLY_TEST_GROUP
                        specifies name to use for read only test group
  --write_only_test_group WRITE_ONLY_TEST_GROUP
                        specifies name to use for write only test group
  --mode MODE           write,read,both
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differentiate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number / serial number of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --csv_outfile CSV_OUTFILE
                        --csv_outfile <Output file for csv data>
  --test_id TEST_ID     test-id for kpi.csv,  script or test name
  --add_to_group ADD_TO_GROUP
                        name of test group to add cxs to
  --del_from_group DEL_FROM_GROUP
                        name of test group to delete cxs from
  --cxs CXS             list of cxs to add/remove depending on use of --add_to_group or --del_from_group

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_bare_argparse found in /lanforge-scripts/py-json/LANfo
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --debug, -d           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80 or https://user:pass@proxy.localnet:3128
  --help_summary        Show summary of what this script does

Creates FileIO endpoints which can be NFS, CIFS or iSCSI endpoints.
```

py-scripts/test_generic.py

```
usage: test_generic.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                       [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                       [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                       [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                       [--proxy [PROXY]]
                       [--debugging DEBUGGING [DEBUGGING ...]]
                       [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
                       [--mode MODE] [--ap AP] [--output_format OUTPUT_FORMAT]
                       [--report_file REPORT_FILE] [--a_min A_MIN]
                       [--b_min B_MIN] [--gen_cols GEN_COLS]
                       [--port_mgr_cols PORT_MGR_COLS]
                       [--compared_report COMPARED_REPORT]
                       [--monitor_interval MONITOR_INTERVAL] [--radio RADIO]
                       [--security SECURITY] [--ssid SSID] [--passwd PASSWD]
                       [--type TYPE] [--cmd CMD] [--dest DEST]
                       [--test_duration TEST_DURATION] [--interval INTERVAL]
                       [--speedtest_min_up SPEEDTEST_MIN_UP]
                       [--speedtest_min_dl SPEEDTEST_MIN_DL]
                       [--speedtest_max_ping SPEEDTEST_MAX_PING]
                       [--client CLIENT] [--file_output FILE_OUTPUT]
                       [--loop_count LOOP_COUNT] [--lf_user LF_USER]
                       [--lf_passwd LF_PASSWD] [--test_rig TEST_RIG]
                       [--test_tag TEST_TAG] [--dut_hw_version DUT_HW_VERSION]
```

```
                      [--dut_sw_version DUT_SW_VERSION]
                      [--dut_model_num DUT_MODEL_NUM]
                      [--dut_serial_num DUT_SERIAL_NUM]
                      [--test_priority TEST_PRIORITY]
                      [--csv_outfile CSV_OUTFILE]

test_generic.py
--------------------
Generic command example:
python3 ./test_generic.py
    --mgr localhost (optional)
    --mgr_port 4122 (optional)
    --upstream_port eth1 (optional)
    --radio wiphy0 (required)
    --num_stations 3 (optional)
    --security {open | wep | wpa | wpa2 | wpa3} (required)
    --ssid netgear (required)
    --passwd admin123 (required)
    --type lfping  {generic | lfping | iperf3-client | speedtest | lf_curl} (required)
    --dest 10.40.0.1 (required - also target for iperf3)
    --test_duration 2m
    --interval 1s
    --debug

    Example commands:
    LFPING:
        ./test_generic.py --mgr localhost --mgr_port 4122 --radio 1.1.wiphy0 --ssid Logan-Test-Net --passwd Logan-Test-Net
        --security wpa2 --num_stations 4 --type lfping --dest 192.168.1.1 --debug --log_level info
        --report_file /home/lanforge/reports/LFPING.csv --test_duration 20s --upstream_port 1.1.eth2
    LFCURL:
        ./test_generic.py --mgr localhost --mgr_port 4122 --radio 1.1.wiphy0 --file_output /home/lanforge/reports/LFCURL.csv
        --num_stations 2 --ssid Logan-Test-Net --passwd Logan-Test-Net --security wpa2 --type lfcurl --dest 192.168.1.1
    GENERIC:
        ./test_generic.py --mgr localhost --mgr_port 4122 --radio 1.1.wiphy0 --num_stations 2 --ssid Logan-Test-Net
        --report_file /home/lanforge/reports/GENERIC.csv --passwd Logan-Test-Net --security wpa2 --type generic
    SPEEDTEST:
        ./test_generic.py --radio 1.1.wiphy0 --num_stations 2 --report_file /home/lanforge/reports/SPEEDTEST.csv
        --ssid Logan-Test-Net --passwd Logan-Test-Net --type speedtest --speedtest_min_up 20 --speedtest_min_dl 20 --speedtest_max
    IPERF3 (under construction):
        ./test_generic.py --mgr localhost --mgr_port 4122 --radio wiphy1 --num_stations 3 --ssid jedway-wpa2-x2048-4-1 --passwd je

optional arguments:
  -h, --help            show this help message and exit
  --type TYPE           type of command to run: generic, lfping, iperf3-client, iperf3-server, lfcurl
  --cmd CMD             specifies command to be run by generic type endp
  --dest DEST           destination IP for command
  --test_duration TEST_DURATION
                        duration of the test eg: 30s, 2m, 4h
  --interval INTERVAL   interval to use when running lfping (1s, 1m)
  --speedtest_min_up SPEEDTEST_MIN_UP
                        sets the minimum upload threshold for the speedtest type
  --speedtest_min_dl SPEEDTEST_MIN_DL
                        sets the minimum download threshold for the speedtest type
  --speedtest_max_ping SPEEDTEST_MAX_PING
                        sets the minimum ping threshold for the speedtest type
  --client CLIENT       client to the iperf3 server
  --file_output FILE_OUTPUT
                        location to output results of lf_curl, absolute path preferred
  --loop_count LOOP_COUNT
                        determines the number of loops to use in lf_curl
  --lf_user LF_USER     user: lanforge
  --lf_passwd LF_PASSWD
                        passwd: lanforge
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differentiate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number / serial number of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --csv_outfile CSV_OUTFILE
                        --csv_outfile <Output file for csv data>

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does
```

```
  --mode MODE           Used to force mode of stations
  --ap AP               Used to force a connection to a particular AP
  --output_format OUTPUT_FORMAT
                        choose either csv or xlsx
  --report_file REPORT_FILE
                        where you want to store results
  --a_min A_MIN         --a_min bps rate minimum for side_a
  --b_min B_MIN         --b_min bps rate minimum for side_b
  --gen_cols GEN_COLS   Columns wished to be monitored from layer 3 endpoint tab
  --port_mgr_cols PORT_MGR_COLS
                        Columns wished to be monitored from port manager tab
  --compared_report COMPARED_REPORT
                        report path and file which is wished to be compared with new report
  --monitor_interval MONITOR_INTERVAL
                        how frequently do you want your monitor function to take measurements; 250ms, 35s, 2h

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

Create generic endpoints and test for their ability to execute chosen commands
```

py-scripts/testgroup.py

```
usage: testgroup.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [-u UPSTREAM_PORT]
                    [--num_stations NUM_STATIONS] [--test_id TEST_ID] [-d]
                    [--log_level LOG_LEVEL]
                    [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                    [--proxy [PROXY]] [--debugging DEBUGGING [DEBUGGING ...]]
                    [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
                    [--radio RADIO] [--security SECURITY] [--ssid SSID]
                    [--passwd PASSWD] [--a_min A_MIN] [--b_min B_MIN]
                    [--mode MODE] [--ap AP] [--group_name GROUP_NAME]
                    [--list_groups] [--show_group]
                    [--add_cx [ADD_CX [ADD_CX ...]]]
                    [--remove_cx [REMOVE_CX [REMOVE_CX ...]]] [--use_existing]
                    [--add_group | --del_group]
                    [--start_group START_GROUP | --stop_group STOP_GROUP | --quiesce_group QUIESCE_GROUP]

NAME: testgroup.py

PURPOSE:
    This script will create a test connection group in the LANforge GUI (Connection Group GUI tab).
     Test Groups are also referred to as Connection Groups.
     The script can preform the following tasks:
     - create a test group
     - add layer-3 cx's to a test group
     - remove layer-3 cx's from a test group
     - start and stop a test connection group

EXAMPLE:
    For an eth-to-eth test connection group (eth-to-eth Layer-3 connection must be manually created first):
        ./testgroup.py --mgr localhost --group_name eth_group --add_group --add_cx l3_eth_test --list_groups

    eth-to-eth JSON command example:
        "args":["--mgr","localhost",
                    "--group_name","eth_group",
                    "--add_group",
                    "--add_cx","l3_eth_test",
                    "--list_groups"
                    ]

    * Add multiple layer-3 cross-connections to a single connection group:
        ./testgroup.py --mgr localhost --group_name group1 --add_group --add_cx l3_test1,l3_test2 --list_groups --use_existing

    * Remove multiple layer-3 cx's from a connection group:
        ./testgroup.py --mgr 192.168.30.12 --group_name group1 --remove_cx l3_test1,l3_test2 --list_groups --use_existing

    * Add a single layer-3 cross connection to a connection group:
        ./testgroup.py --mgr localhost --group_name group1 --add_group --add_cx l3_test --list_groups --use_existing

    * Remove a layer-3 cx from a specified connection group:
        ./testgroup.py --mgr localhost --group_name group1 --remove_cx l3_test1 --list_groups --use_existing

    * Add single layer-3 cross-connections to a single connection group and strat group:

        ./testgroup.py --mgr 192.168.200.93 --group_name CX_GROUP --add_group --add_cx l3_test --start_group CX_GROUP --use_existi

    * Add multiple layer-3 cross-connections to a single connection group and strat group:

        ./testgroup.py --mgr 192.168.200.93 --group_name CX_GROUP --add_group --add_cx l3_test,l3_test1 --start_group CX_GROUP --u

    * Start Selected Group:

        ./testgroup.py --mgr 192.168.200.93 --group_name CX_GROUP --start_group CX_GROUP --use_existing

    * Stop Selected Group:

        ./testgroup.py --mgr 192.168.200.93 --group_name CX_GROUP --stop_group CX_GROUP --use_existing

    * Quiesce Selected Group:

        ./testgroup.py --mgr 192.168.200.93 --group_name CX_GROUP --quiesce_group CX_GROUP --use_existing

    * Delete Selected Group:

        ./testgroup.py --mgr 192.168.200.93 --group_name CX_GROUP1 --del_group --use_existing


    * To create given number of stations and l3 cross-connections along with add them in a test-group.

         ./testgroup.py --mgr 192.168.200.138 --num_stations 2 --ssid Netgear2g --passwd lanforge --security wpa2
         --radio wiphy0 --group_name group0 --add_group --upstream_port eth2 --a_min 6000 --b_min 6000
```

```
    * To create given number of stations and l3 cross-connections along with add them in a test-group & Start Selected Group:

        ./testgroup.py --mgr 192.168.200.138 --num_stations 2 --ssid Netgear2g --passwd lanforge --security wpa2
        --radio wiphy0 --group_name group0 --start_group group0

SCRIPT_CLASSIFICATION:  Creation, Addition, Deletion, Creation stations & test-groups

SCRIPT_CATEGORIES: Functional

STATUS: Functional

VERIFIED_ON:   7-AUG-2023,
            GUI Version:  5.4.6
            Kernel Version: 6.2.16+

LICENSE:
          Free to distribute and modify. LANforge systems must be licensed.
          Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False


optional arguments:
  -h, --help            show this help message and exit
  --a_min A_MIN         --a_min bps rate minimum for side_a
  --b_min B_MIN         --b_min bps rate minimum for side_b
  --mode MODE           Used to force mode of stations
  --ap AP               Used to force a connection to a particular AP
  --group_name GROUP_NAME
                        specify the name of the test group to use
  --list_groups         list all existing test groups
  --show_group          show connections in current test group
  --add_cx [ADD_CX [ADD_CX ...]]
                        add cx to chosen test group
  --remove_cx [REMOVE_CX [REMOVE_CX ...]]
                        remove cx from chosen test group
  --use_existing        specify this to use already existed cx for connection group.
  --add_group           add new test group
  --del_group           delete test group
  --start_group START_GROUP
                        start all cxs in chosen test group
  --stop_group STOP_GROUP
                        stop all cxs in chosen test group
  --quiesce_group QUIESCE_GROUP
                        quiesce all cxs in chosen test groups

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

Control and query test groups
```

py-scripts/test_ip_connection.py

```
usage: test_ip_connection.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                             [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                             [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                             [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                             [--proxy [PROXY]]
                             [--debugging DEBUGGING [DEBUGGING ...]]
                             [--debug_log DEBUG_LOG] [--no_cleanup]
                             [--help_summary] [--radio RADIO]
                             [--security SECURITY] [--ssid SSID]
                             [--passwd PASSWD] [--ipv6] [--ap AP]
                             [--mode MODE] [--timeout TIMEOUT]
                             [--use_existing_sta]

        test_ip_connection.py
--------------------------------------
Generic ipv6 command example:
python3 ./test_ip_connection.py
        --upstream_port eth1
        --radio wiphy0
```

```
        --num_stations 3
        --ipv6
        --proxy
        --security {open|wep|wpa|wpa2|wpa3}
        --ssid netgear
        --passwd admin123
        --mode   1
        --ap "00:0e:8e:78:e1:76"
        --test_id
        --timeout 120
        --debug

Generic ipv4 command example:
./test_ip_connection.py
    --upstream_port eth1
    --radio wiphy0
    --num_stations 3
    --security open
    --ssid netgear
    --passwd BLANK
    --debug

optional arguments:
  -h, --help            show this help message and exit
  --ipv6                Use ipv6 connections instead of ipv4
  --ap AP               Add BSSID of access point to connect to
  --mode MODE           Station WiFi modes: use the number value below:
                                        auto   : 0,
                                        a      : 1,
                                        b      : 2,
                                        g      : 3,
                                        abg    : 4,
                                        abgn   : 5,
                                        bgn    : 6,
                                        bg     : 7,
                                        abgnAC : 8,
                                        anAC   : 9,
                                        an     : 10,
                                        bgnAC  : 11,
                                        abgnAX : 12,
                                        bgnAX  : 13
  --timeout TIMEOUT     --timeout sets the length of time to wait until a connection is successful
  --use_existing_sta    Used an existing stationsto a particular AP

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

        Create stations that attempt to authenticate, associate, and receive IP addresses on the
        chosen SSID
```

py-scripts/test_ipv4_ps.py

```
usage: test_ipv4_variable_time.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                                  [-u UPSTREAM_PORT]
                                  [--num_stations NUM_STATIONS]
                                  [--test_id TEST_ID] [-d]
                                  [--log_level LOG_LEVEL]
                                  [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                  [--proxy [PROXY]]
                                  [--debugging DEBUGGING [DEBUGGING ...]]
                                  [--debug_log DEBUG_LOG] [--no_cleanup]
                                  [--help_summary] [--radio RADIO]
                                  [--security SECURITY] [--ssid SSID]
                                  [--passwd PASSWD] [--a_min A_MIN]
                                  [--b_min B_MIN]
                                  [--test_duration TEST_DURATION]
                                  [--radio2 RADIO2]

test_ipv4_variable_time.py:
---------------------

Generic command layout:
```

```
./test_ipv4_variable_time.py
--upstream_port eth1
--radio wiphy3
--num_stations 4
--ssid jedway-wpa2-x2048-4-1
--passwd jedway-wpa2-x2048-4-1
--security  {wpa2|open|wpa|wpa3}
--a_min 250000
--b_min 260000
--test_duration 2m
--debug


optional arguments:
  -h, --help            show this help message and exit
  --a_min A_MIN         --a_min bps rate minimum for side_a
  --b_min B_MIN         --b_min bps rate minimum for side_b
  --test_duration TEST_DURATION
                        --test_duration sets the duration of the test
  --radio2 RADIO2       radio to create monitor on

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key
```

py-scripts/test_ipv4_ttls.py

```
usage: test_ipv4_ttls.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                         [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                         [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                         [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                         [--proxy [PROXY]]
                         [--debugging DEBUGGING [DEBUGGING ...]]
                         [--debug_log DEBUG_LOG] [--no_cleanup]
                         [--help_summary] [--radio RADIO]
                         [--security SECURITY] [--ssid SSID] [--passwd PASSWD]
                         [--a_min A_MIN] [--b_min B_MIN]
                         [--test_duration TEST_DURATION] [--key-mgmt KEY_MGMT]
                         [--wpa_psk WPA_PSK] [--eap EAP] [--identity IDENTITY]
                         [--ttls_passwd TTLS_PASSWD] [--ttls_realm TTLS_REALM]
                         [--domain DOMAIN] [--hessid HESSID]
                         [--ieee80211w IEEE80211W] [--use_hs20 USE_HS20]
                         [--enable_pkc ENABLE_PKC] [--vap VAP]

test_ipv4_ttls.py:
 --------------------
 Generic command layout:
 python ./test_ipv4_ttls.py

    --upstream_port eth1
    --radio wiphy0
    --num_stations 3
    --ssid ssid-wpa-1
    --key ssid-wpa-1
    --security <security type: wpa2, open, wpa, wpa3>
    --debug

optional arguments:
  -h, --help            show this help message and exit
  --a_min A_MIN         --a_min bps rate minimum for side_a
  --b_min B_MIN         --b_min bps rate minimum for side_b
  --test_duration TEST_DURATION
                        --test_duration sets the duration of the test
  --key-mgmt KEY_MGMT   --key-mgt: { DEFAULT, NONE, WPA-PSK, FT-PSK, FT-EAP, FT-SAE, FT-EAP-SHA384, WPA-EAP, OSEN, IEEE8021X, WPA-
  --wpa_psk WPA_PSK     wpa-ent pre shared key
  --eap EAP             --eap eap method to use
  --identity IDENTITY   --identity eap identity string
  --ttls_passwd TTLS_PASSWD
                        --ttls_passwd eap password string
  --ttls_realm TTLS_REALM
                        --ttls_realm 802.11u home realm to use
  --domain DOMAIN       --domain 802.11 domain to use
  --hessid HESSID       --hessid 802.11u HESSID (MAC addr format/peer for WDS)
  --ieee80211w IEEE80211W
```

```
                        --ieee80211w <disabled(0),optional(1),required(2)
  --use_hs20 USE_HS20   use HotSpot 2.0
  --enable_pkc ENABLE_PKC
                        enable opportunistic PMKSA WPA2 key caching
  --vap VAP             Create VAP on host device

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

Demonstration showing wpa2-ent ttls authentication
```

py-scripts/test_ip_variable_time.py

```
usage: test_ip_variable_time.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                                [-u UPSTREAM_PORT [UPSTREAM_PORT ...]]
                                [--num_stations NUM_STATIONS]
                                [--test_id TEST_ID] [-d]
                                [--log_level LOG_LEVEL]
                                [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                [--proxy [PROXY]]
                                [--debugging DEBUGGING [DEBUGGING ...]]
                                [--debug_log DEBUG_LOG] [--no_cleanup]
                                [--radio RADIO [RADIO ...]]
                                [--security SECURITY [SECURITY ...]]
                                [--ssid SSID [SSID ...]]
                                [--passwd PASSWD [PASSWD ...]]
                                [--mode MODE [MODE ...]] [--ap AP [AP ...]]
                                [--traffic_type TRAFFIC_TYPE]
                                [--output_format OUTPUT_FORMAT]
                                [--report_file REPORT_FILE] [--a_min A_MIN]
                                [--b_min B_MIN]
                                [--test_duration TEST_DURATION]
                                [--layer3_cols LAYER3_COLS]
                                [--port_mgr_cols PORT_MGR_COLS]
                                [--compared_report COMPARED_REPORT]
                                [--monitor_interval MONITOR_INTERVAL] [--ipv6]
                                [--use_existing_sta] [--sta_names STA_NAMES]
                                [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                                [--resource RESOURCE] [--test_rig TEST_RIG]
                                [--test_tag TEST_TAG]
                                [--dut_hw_version DUT_HW_VERSION]
                                [--dut_sw_version DUT_SW_VERSION]
                                [--dut_model_num DUT_MODEL_NUM]
                                [--dut_serial_num DUT_SERIAL_NUM]
                                [--test_priority TEST_PRIORITY]
                                [--csv_outfile CSV_OUTFILE] [--help_summary]

NAME: test_ip_variable_time.py

PURPOSE:
          ->  This script creates variable number of stations with individual cross-connects and endpoints.
              Stations are set to UP state, but cross-connections remain stopped.

          ->  This script create variable number of stations and endpoints to generate and verify layer-3 traffic over ipv4 or ipv

          ->  Create stations to test connection and traffic on VAPs of varying security types (WEP, WPA, WPA2, WPA3, Open)
              over ipv4 or ipv6

EXAMPLE:
    1. Use Existing station ,  Note: put the shelf.resource.wifi-sta  (below is 1.1.wlan4),
        The station needs to configured with the ssid, passwd, security and mode in the LANforge GUI
    ./test_ip_variable_time.py  --mgr 192.168.0.100  --radio wiphy4 --ssid ssid_5g --passwd pass_5g
        --security wpa2 --test_duration 60s --output_format csv  --traffic_type lf_tcp
        --a_min 600000000 --b_min 600000000  --upstream_port eth2 --mode '5'
        --layer3_cols 'name','tx rate','rx rate'  --port_mgr_cols 'alias','channel','activity','mode'
        --use_existing_sta --sta_names 1.1.wlan4

    2. Create a one station (script default is 1 if --num_stations not entered)
    ./test_ip_variable_time.py  --mgr 192.168.0.100    --radio wiphy6 --ssid ssid_5g --passwd pass_5g
        --security wpa2 --test_duration 60s --output_format csv  --traffic_type lf_tcp
        --a_min 600000000 --b_min 600000000  --upstream_port eth2 --mode '5'
        --layer3_cols 'name','tx rate','rx rate'  --port_mgr_cols 'alias','channel','activity','mode'
```

```
    3. Create two stations
    ./test_ip_variable_time.py  --mgr 192.168.0.100    --radio wiphy1 --ssid ssid_5g --passwd pass_5g
        --security wpa2 --test_duration 60s --output_format csv  --traffic_type lf_tcp
        --a_min 600000000 --b_min 600000000  --upstream_port eth2 --mode '5'
        --layer3_cols 'name','tx rate','rx rate'  `--port_mgr_cols 'alias','channel','activity','mode'
        --num_stations 2

    4. Create Multiple stations and run traffic with different upstream port
    ./test_ip_variable_time.py --mgr 192.168.200.37  --radio wiphy0 wiphy0 --ssid ssid_2g ssid_5g
        --test_duration 60s --output_format csv  --traffic_type lf_tcp --a_min 600000000 --b_min 600000000
        --upstream_port eth2 eth1 --mode '5' --num_stations 1 --passwd pass_2g pass_5g --security wpa2 wpa2

SCRIPT_CLASSIFICATION:  Creation & Runs Traffic

SCRIPT_CATEGORIES:  Performance, Functional,  KPI Generation,  Report Generation

NOTES:
      * This Script has two working modes:

        Mode 1:
            When station is not available,

            This script will create a variable number of stations each with their own set of cross-connects and endpoints.
            It will then create layer 3 traffic over a specified amount of time, testing for increased traffic at regular interval
            This test  will pass if all stations increase traffic over the full test duration.

        Mode 2:
            When station is already available,

            This script will create layer3 cross-connects and endpoints It will then
            create layer 3 traffic over a specified amount of time, testing for increased traffic at regular intervals.
            This test will pass if all stations increase traffic over the full test duration.

        Generic command layout:

        python3 ./test_ip_variable_time.py
            --upstream_port eth1
            --radio wiphy0
            --num_stations 32
            --security {open|wep|wpa|wpa2|wpa3}
            --mode   1
                {"auto"   : "0",
                "a"      : "1",
                "b"      : "2",
                "g"      : "3",
                "abg"    : "4",
                "abgn"   : "5",
                "bgn"    : "6",
                "bg"     : "7",
                "abgnAC" : "8",
                "anAC"   : "9",
                "an"     : "10",
                "bgnAC"  : "11",
                "abgnAX" : "12",
                "bgnAX"  : "13"}
            --ssid <ssid>
            --password admin123
            --test_duration 2m (default)
            --monitor_interval
            --a_min 3000
            --b_min 1000
            --ap "00:0e:8e:78:e1:76"
            --output_format csv
            --traffic_type lf_udp
            --report_file ~/Documents/results.csv               (Example of csv file output  - please use another extension for ot
            --compared_report ~/Documents/results_prev.csv      (Example of csv file retrieval - please use another extension for
            --layer3_cols 'name','tx bytes','rx bytes','dropped'(column names from the GUI to print on report -  please read below
            --port_mgr_cols 'ap','ip'                           (column names from the GUI to print on report -  please read below
            --debug

        ================================================================================
        ** FURTHER INFORMATION **
            Using the layer3_cols flag:

            Currently the output function does not support inputting the columns in layer3_cols the way they are displayed in the
            This quirk is under construction. To output certain columns in the GUI in your final report, please match the accordin
            GUI column displayed to it's counterpart to have the additional columns correctly displayed in your report.
            Note that the report will prepend "l3-" to the supplied layer3_col flags.

            GUI Column Display       Layer3_cols argument to type in (to print in report)

            Name                |  'name'
            EID                 |  'eid'
            Run                 |  'run'
            Mng                 |  'mng'
            Script              |  'script'
            Tx Rate             |  'tx rate'
            Tx Rate (1 min)     |  'tx rate (1 min)'
            Tx Rate (last)      |  'tx rate (last)'
            Tx Rate LL          |  'tx rate ll'
            Rx Rate             |  'rx rate'
            Rx Rate (1 min)     |  'rx rate (1 min)'
            Rx Rate (last)      |  'rx rate (last)'
            Rx Rate LL          |  'rx rate ll'
            Rx Drop %           |  'rx drop %'
            Tx PDUs             |  'tx pdus'
            Tx Pkts LL          |  'tx pkts ll'
            PDU/s TX            |  'pdu/s tx'
            Pps TX LL           |  'pps tx ll'
            Rx PDUs             |  'rx pdus'
            Rx Pkts LL          |  'pps rx ll'
            PDU/s RX            |  'pdu/s tx'
            Pps RX LL           |  'pps rx ll'
            Delay               |  'delay'
            Dropped             |  'dropped'
            Jitter              |  'jitter'
            Tx Bytes            |  'tx bytes'
```

```
Rx Bytes            |  'rx bytes'
Replays             |  'replays'
TCP Rtx             |  'tcp rtx'
Dup Pkts            |  'dup pkts'
Rx Dup %            |  'rx dup %'
OOO Pkts            |  'ooo pkts'
Rx OOO %            |  'rx ooo %'
RX Wrong Dev        |  'rx wrong dev'
CRC Fail            |  'crc fail'
RX BER              |  'rx ber'
CX Active           |  'cx active'
CX Estab/s          |  'cx estab/s'
1st RX              |  '1st rx'
CX TO               |  'cx to'
Pattern             |  'pattern'
Min PDU             |  'min pdu'
Max PDU             |  'max pdu'
Min Rate            |  'min rate'
Max Rate            |  'max rate'
Send Buf            |  'send buf'
Rcv Buf             |  'rcv buf'
CWND                |  'cwnd'
TCP MSS             |  'tcp mss'
Bursty              |  'bursty'
A/B                 |  'a/b'
Elapsed             |  'elapsed'
Destination Addr    |  'destination addr'
Source Addr         |  'source addr'

Using the port_mgr_cols flag:
    '4way time (us)'
    'activity'
    'alias'
    'anqp time (us)'
    'ap'
    'beacon'
    'bps rx'
    'bps rx ll'
    'bps tx'
    'bps tx ll'
    'bytes rx ll'
    'bytes tx ll'
    'channel'
    'collisions'
    'connections'
    'crypt'
    'cx ago'
    'cx time (us)'
    'device'
    'dhcp (ms)'
    'down'
    'entity id'
    'gateway ip'
    'ip'
    'ipv6 address'
    'ipv6 gateway'
    'key/phrase'
    'login-fail'
    'login-ok'
    'logout-fail'
    'logout-ok'
    'mac'
    'mask'
    'misc'
    'mode'
    'mtu'
    'no cx (us)'
    'noise'
    'parent dev'
    'phantom'
    'port'
    'port type'
    'pps rx'
    'pps tx'
    'qlen'
    'reset'
    'retry failed'
    'rx bytes'
    'rx crc'
    'rx drop'
    'rx errors'
    'rx fifo'
    'rx frame'
    'rx length'
    'rx miss'
    'rx over'
    'rx pkts'
    'rx-rate'
    'sec'
    'signal'
    'ssid'
    'status'
    'time-stamp'
    'tx abort'
    'tx bytes'
    'tx crr'
    'tx errors'
    'tx fifo'
    'tx hb'
    'tx pkts'
    'tx wind'
    'tx-failed %'
    'tx-rate'
    'wifi retries'

Can't decide what columns to use? You can just use 'all' to select all available columns from both tables.
```

```
           This script uses two args parsers one in the script the second is Realm args parser
           Realm args parser is one directory up then traverse into /py-json/LANforge/lfcli_base.py
           search for create_basic_argsparse
           --mgr --mgr_port --upstream_port --num_stations --radio --security --ssid --passwd

STATUS: Functional

VERIFIED_ON:   15-JULY-2023,
            Build Version:  5.4.6
            Kernel Version: 6.2.16+

LICENSE:
         Free to distribute and modify. LANforge systems must be licensed.
         Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False



optional arguments:
  -h, --help            show this help message and exit
  --use_existing_sta    Used an existing stations to a particular AP
  --sta_names STA_NAMES
                        Used to force a connection to a particular AP
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path, primary use when running test in test suite
  --resource RESOURCE   LANforge Station resource ID to use, default is 1
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differenciate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number / serial number of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --csv_outfile CSV_OUTFILE
                        --csv_outfile <Output file for csv data>
  --help_summary        shows help summary of the script

optional arguments:
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT [UPSTREAM_PORT ...], --upstream_port UPSTREAM_PORT [UPSTREAM_PORT ...]
                        non-station port that generates traffic: <resource>.<port>, e.g: --u eth1 eth2
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --mode MODE [MODE ...]
                        Used to force mode of stations e.g: --mode 11 9
  --ap AP [AP ...]      Used to force a connection to a particular AP
  --traffic_type TRAFFIC_TYPE
                        Select the Traffic Type [lf_udp, lf_tcp, udp, tcp], type will be adjusted automatically between ipv4 and i
  --output_format OUTPUT_FORMAT
                        choose either csv or xlsx
  --report_file REPORT_FILE
                        where you want to store results
  --a_min A_MIN         --a_min bps rate minimum for side_a
  --b_min B_MIN         --b_min bps rate minimum for side_b
  --test_duration TEST_DURATION
                        --test_duration sets the duration of the test
  --layer3_cols LAYER3_COLS
                        Additional columns wished to be monitored from the layer 3 endpoint tab for reporting
  --port_mgr_cols PORT_MGR_COLS
                        Additional columns wished to be monitored from port manager tab for reporting
  --compared_report COMPARED_REPORT
                        report path and file which is wished to be compared with new report
  --monitor_interval MONITOR_INTERVAL
                        how frequently do you want your monitor function to take measurements, 35s, 2h, lowest is 250ms
  --ipv6                Sets the test to use IPv6 traffic instead of IPv4

required arguments:
  --radio RADIO [RADIO ...]
                        radio EID, e.g: --radio wiphy0 wiphy2
  --security SECURITY [SECURITY ...]
                        WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >  e.g: --security open wpa
  --ssid SSID [SSID ...]
                        WiFi SSID for script objects to associate to e.g: --ssid ap1_ssid ap2_ssid
  --passwd PASSWD [PASSWD ...], --password PASSWD [PASSWD ...], --key PASSWD [PASSWD ...]
                        WiFi passphrase/password/key e.g: --passwd [BLANK] passwd@123

           Create stations to test connection and traffic on VAPs of varying security types (WEP, WPA, WPA2, WPA3, Open)
           over ipv4 or ipv6
```

```
usage: test_l3_longevity.py [-h] [--help_summary]
                            [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                            [--test_rig TEST_RIG] [--test_tag TEST_TAG]
                            [--dut_hw_version DUT_HW_VERSION]
                            [--dut_sw_version DUT_SW_VERSION]
                            [--dut_model_num DUT_MODEL_NUM]
                            [--dut_serial_num DUT_SERIAL_NUM]
                            [--test_priority TEST_PRIORITY]
                            [--test_id TEST_ID] [-o CSV_OUTFILE] [--tty TTY]
                            [--baud BAUD] [--mgr LFMGR]
                            [--mgr_port LFMGR_PORT]
                            [--test_duration TEST_DURATION] [--tos TOS]
                            [--debug] [-t ENDP_TYPE] [-u UPSTREAM_PORT]
                            [--downstream_port DOWNSTREAM_PORT]
                            [--polling_interval POLLING_INTERVAL] [-r RADIO]
                            [--collect_layer3_data] [--ap_read]
                            [--ap_scheme {serial,telnet,ssh,mux_serial}]
                            [--ap_port AP_PORT] [--ap_baud AP_BAUD]
                            [--ap_ip AP_IP] [--ap_ssh_port AP_SSH_PORT]
                            [--ap_user AP_USER] [--ap_passwd AP_PASSWD]
                            [--ap_if_2g AP_IF_2G] [--ap_if_5g AP_IF_5G]
                            [--ap_if_6g AP_IF_6G] [--ap_cmd_6g AP_CMD_6G]
                            [--ap_cmd_5g AP_CMD_5G] [--ap_cmd_2g AP_CMD_2G]
                            [--ap_cmd_ul_6g AP_CMD_UL_6G]
                            [--ap_cmd_ul_5g AP_CMD_UL_5G]
                            [--ap_cmd_ul_2g AP_CMD_UL_2G]
                            [--ap_chanim_cmd_6g AP_CHANIM_CMD_6G]
                            [--ap_chanim_cmd_5g AP_CHANIM_CMD_5G]
                            [--ap_chanim_cmd_2g AP_CHANIM_CMD_2G]
                            [--ap_scheduler_stats] [--ap_ofdma_stats]
                            [--ap_test_mode] [-amr SIDE_A_MIN_BPS]
                            [-amp SIDE_A_MIN_PDU] [-bmr SIDE_B_MIN_BPS]
                            [-bmp SIDE_B_MIN_PDU] [--rates_are_totals]
                            [--multiconn MULTICONN]
                            [--attenuators ATTENUATORS]
                            [--atten_vals ATTEN_VALS] [--cap_ctl_out]
                            [--wait WAIT]
                            [--sta_start_offset STA_START_OFFSET]
                            [--no_pre_cleanup] [--no_cleanup]
                            [--no_stop_traffic] [--use_existing_station_list]
                            [--existing_station_list EXISTING_STATION_LIST]
                            [--wait_for_ip_sec WAIT_FOR_IP_SEC]
                            [--log_level LOG_LEVEL]
                            [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]

test_l3_longevity.py:
--------------------

Summary :
----------
The Layer 3 Traffic Generation Test is designed to test the performance of the
Access Point by running layer 3 Cross-Connect Traffic.  Layer-3 Cross-Connects represent a stream
of data flowing through the system under test. A Cross-Connect (CX) is composed of two Endpoints,
each of which is associated with a particular Port (physical or virtual interface).

The test will create stations, create cx traffic between upstream port and stations, run traffic
and generate a report.

Generic command layout:
-----------------------
./test_l3_longevity.py --mgr <ip_address> --test_duration <duration> --endp_type <traffic types>
--upstream_port <shelf>.<resource>.<port>
--radio "radio=<shelf>.<resource>.<radio> stations==<number stations> ssid==<ssid> ssid_pw==<ssid password>
security==<security type: wpa2, open, wpa3>" --debug

Multiple radios may be entered with individual --radio switches

# UDP bi-directional test, no use of controller.
./test_l3_longevity.py --mgr localhost --endp_type 'lf_udp lf_tcp' --upstream_port 1.1.eth1
--radio "radio==1.1.wiphy0 stations==10 ssid==ASUS_70 ssid_pw==[BLANK] security==open"
--radio "radio==1.1.wiphy2 stations==1 ssid==ASUS_70 ssid_pw==[BLANK] security==open" --test_duration 30s

# Port resets, chooses random value between min and max
./test_l3_longevity.py --lfmgr LF_MGR_IP --test_duration 90s --polling_interval 10s --upstream_port 1.1.eth2
--radio 'radio==1.1.wiphy1,stations==4,ssid==SSID_USED,ssid_pw==SSID_PW_USED,security==SECURITY_USED,
reset_port_enable==TRUE,reset_port_time_min==10s,reset_port_time_max==20s'
--endp_type lf_udp --rates_are_totals --side_a_min_bps=20000 --side_b_min_bps=300000000"

<duration>: number followed by one of the following
d - days
h - hours
m - minutes
s - seconds

<traffic type>:
lf_udp  : IPv4 UDP traffic
lf_tcp  : IPv4 TCP traffic
lf_udp6 : IPv6 UDP traffic
lf_tcp6 : IPv6 TCP traffic
mc_udp  : IPv4 multi cast UDP traffic
mc_udp6 : IPv6 multi cast UDP traffic

<tos>:
BK, BE, VI, VO:  Optional wifi related Tos Settings.  Or, use your preferred numeric values.

################################
# Command switches
################################

--mgr <hostname for where LANforge GUI is running>',default='localhost'
-d  / --test_duration <how long to run>  example --time 5d (5 days) default: 3m options: number followed by d, h, m or s',default=
--tos:  Support different ToS settings: BK | BE | VI | VO | numeric',default="BE"
--debug:  Enable debugging',default=False
-t  / --endp_type <types of traffic> example --endp_type "lf_udp lf_tcp mc_udp"  Default: lf_udp , options: lf_udp, lf_udp6, lf_tc
                        default='lf_udp', type=valid_endp_types
```

```
-u / --upstream_port <cross connect upstream_port> example: --upstream_port eth1',default='eth1')
-o / --outfile <Output file for csv data>", default='longevity_results'

########################################
# Examples
# ######################################
Example #1  running traffic with two radios
1. Test duration 30 minutes
2. Traffic IPv4 TCP, UDP
3. Upstream-port eth2
4. Radio #0 wiphy0 has 1 station, ssid = ssid_2g, ssid password = ssid_pw_2g  security = wpa2
5. Radio #1 wiphy1 has 2 stations, ssid = ssid_5g, ssid password = BLANK security = open

Command:
python3 ./test_l3_longevity.py --test_duration 30s --endp_type "lf_tcp lf_udp" --tos "BK VI" --upstream_port eth2
--radio "radio==wiphy0 stations==1 ssid==ssid_2g ssid_pw==ssid_pw_2g security==wpa2"
--radio "radio==wiphy1 stations==2 ssid==ssid_5g ssid_pw==BLANK security==open"

Example : Have the stations continue to run after the completion of the script
    ./test_l3_longevity.py --lfmgr 192.168.0.101 --endp_type 'lf_udp,lf_tcp' --tos BK --upstream_port 1.1.eth2
        --radio 'radio==wiphy1 stations==2 ssid==asus_2g ssid_pw==lf_asus_2g security==wpa2'
        --test_duration 30s --polling_interval 5s
        --side_a_min_bps 256000 --side_b_min_bps 102400000
        --no_stop_traffic

Example : Have script use existing stations from previous run where traffic was not stopped and also create new stations and
        leave traffic running
        ./test_l3_longevity.py --lfmgr 192.168.0.101 --endp_type 'lf_udp,lf_tcp' --tos BK --upstream_port 1.1.eth2
        --radio 'radio==wiphy0 stations==2 ssid==asus_5g ssid_pw==lf_asus_5g security==wpa2'
        --sta_start_offset 1000
        --test_duration 30s --polling_interval 5s
        --side_a_min_bps 256000 --side_b_min_bps 102400000
        --use_existing_station_list
        --existing_station_list '1.1.sta0000,1.1.sta0001'
        --no_stop_traffic

Example : Add the following switches to use ssh to access ASUS (both ssh and serial supported), the interfaces need to be provided

                --ap_read
                --ap_scheme ssh
                --ap_ip 192.168.50.1
                --ap_ssh_port 1025
                --ap_user lanforge
                --ap_passwd lanforge
                --ap_if_2g eth6
                --ap_if_5g eth7
                --ap_if_6g eth8

Setting wifi_settings per radio
./test_l3_longevity.py  --lfmgr 192.168.100.116 --local_lf_report_dir /home/lanforge/html-reports/ --test_duration 15s
--polling_interval 5s --upstream_port 1.1.eth2
--radio "radio==1.1.wiphy1 stations==4 ssid==asus11ax-5 ssid_pw==hello123 security==wpa2  mode==0 wifi_settings==wifi_settings
    enable_flags==('ht160_enable'|'wpa2_enable'|'80211u_enable'|'create_admin_down'|'ht160_enable') "
--endp_type lf_udp --rates_are_totals --side_a_min_bps=20000 --side_b_min_bps=300000000 --test_rig CT-US-001 --test_tag 'l3_longev

Note: for enable flags can us && as separator in vscode

        wifi_mode
        Input       : Enum Val  : Shown by nc_show_ports

        AUTO        |  0        #  802.11g
        802.11a     |  1        #  802.11a
        b           |  2        #  802.11b
        g           |  3        #  802.11g
        abg         |  4        #  802.11abg
        abgn        |  5        #  802.11abgn
        bgn         |  6        #  802.11bgn
        bg          |  7        #  802.11bg
        abgnAC      |  8        #  802.11abgn-AC
        anAC        |  9        #  802.11an-AC
        an          | 10        #  802.11an
        bgnAC       | 11        #  802.11bgn-AC
        abgnAX      | 12        #  802.11abgn-AX
                                #     a/b/g/n/AC/AX (dual-band AX) support
        bgnAX       | 13        #  802.11bgn-AX
        anAX        | 14        #  802.11an-AX
        aAX         | 15        #  802.11a-AX (6E disables /n and /ac)

        wifi_settings flags are currently defined as:
        wpa_enable           | 0x10         # Enable WPA
        custom_conf          | 0x20         # Use Custom wpa_supplicant config file.
        wep_enable           | 0x200        # Use wpa_supplicant configured for WEP encryption.
        wpa2_enable          | 0x400        # Use wpa_supplicant configured for WPA2 encryption.
        ht40_disable         | 0x800        # Disable HT-40 even if hardware and AP support it.
        scan_ssid            | 0x1000       # Enable SCAN-SSID flag in wpa_supplicant.
        passive_scan         | 0x2000       # Use passive scanning (don't send probe requests).
        disable_sgi          | 0x4000       # Disable SGI (Short Guard Interval).
        lf_sta_migrate       | 0x8000       # OK-To-Migrate (Allow station migration between LANforge radios)
        verbose              | 0x10000      # Verbose-Debug:  Increase debug info in wpa-supplicant and hostapd logs.
        80211u_enable        | 0x20000      # Enable 802.11u (Interworking) feature.
        80211u_auto          | 0x40000      # Enable 802.11u (Interworking) Auto-internetworking feature.  Always enabled currentl
        80211u_gw            | 0x80000      # AP Provides access to internet (802.11u Interworking)
        80211u_additional    | 0x100000     # AP requires additional step for access (802.11u Interworking)
        80211u_e911          | 0x200000     # AP claims emergency services reachable (802.11u Interworking)
        80211u_e911_unauth   | 0x400000     # AP provides Unauthenticated emergency services (802.11u Interworking)
        hs20_enable          | 0x800000     # Enable Hotspot 2.0 (HS20) feature.  Requires WPA-2.
        disable_gdaf         | 0x1000000    # AP:  Disable DGAF (used by HotSpot 2.0).
        8021x_radius         | 0x2000000    # Use 802.1x (RADIUS for AP).
        80211r_pmska_cache   | 0x4000000    # Enable oportunistic PMSKA caching for WPA2 (Related to 802.11r).
        disable_ht80         | 0x8000000    # Disable HT80 (for AC chipset NICs only)
        ibss_mode            | 0x20000000   # Station should be in IBSS mode.
        osen_enable          | 0x40000000   # Enable OSEN protocol (OSU Server-only Authentication)
        disable_roam         | 0x80000000   # Disable automatic station roaming based on scan results.
        ht160_enable         | 0x100000000  # Enable HT160 mode.
        disable_fast_reauth  | 0x200000000  # Disable fast_reauth option for virtual stations.
        mesh_mode            | 0x400000000  # Station should be in MESH mode.
        power_save_enable    | 0x800000000  # Station should enable power-save.  May not work in all drivers/configurations.
```

```
       create_admin_down    | 0x1000000000 # Station should be created admin-down.
       wds-mode             | 0x2000000000 # WDS station (sort of like a lame mesh), not supported on ath10k
       no-supp-op-class-ie  | 0x4000000000 # Do not include supported-oper-class-IE in assoc requests.  May work around AP bugs.
       txo-enable           | 0x8000000000 # Enable/disable tx-offloads, typically managed by set_wifi_txo command
       use-wpa3             | 0x10000000000 # Enable WPA-3 (SAE Personal) mode.
       use-bss-transition   | 0x80000000000 # Enable BSS transition.
       disable-twt          | 0x100000000000 # Disable TWT mode

================================================================================
** FURTHER INFORMATION **
   Using the layer3_cols flag:

   Currently the output function does not support inputting the columns in layer3_cols the way they are displayed in the GUI. Thi
   certain columns in the GUI in your final report, please match the according GUI column display to it's counterpart to have the
   your report.

   GUI Column Display       Layer3_cols argument to type in (to print in report)

   Name                |  'name'
   EID                 |  'eid'
   Run                 |  'run'
   Mng                 |  'mng'
   Script              |  'script'
   Tx Rate             |  'tx rate'
   Tx Rate (1 min)     |  'tx rate (1 min)'
   Tx Rate (last)      |  'tx rate (last)'
   Tx Rate LL          |  'tx rate ll'
   Rx Rate             |  'rx rate'
   Rx Rate (1 min)     |  'rx rate (1 min)'
   Rx Rate (last)      |  'rx rate (last)'
   Rx Rate LL          |  'rx rate ll'
   Rx Drop %           |  'rx drop %'
   Tx PDUs             |  'tx pdus'
   Tx Pkts LL          |  'tx pkts ll'
   PDU/s TX            |  'pdu/s tx'
   Pps TX LL           |  'pps tx ll'
   Rx PDUs             |  'rx pdus'
   Rx Pkts LL          |  'pps rx ll'
   PDU/s RX            |  'pdu/s tx'
   Pps RX LL           |  'pps rx ll'
   Delay               |  'delay'
   Dropped             |  'dropped'
   Jitter              |  'jitter'
   Tx Bytes            |  'tx bytes'
   Rx Bytes            |  'rx bytes'
   Replays             |  'replays'
   TCP Rtx             |  'tcp rtx'
   Dup Pkts            |  'dup pkts'
   Rx Dup %            |  'rx dup %'
   OOO Pkts            |  'ooo pkts'
   Rx OOO %            |  'rx ooo %'
   RX Wrong Dev        |  'rx wrong dev'
   CRC Fail            |  'crc fail'
   RX BER              |  'rx ber'
   CX Active           |  'cx active'
   CX Estab/s          |  'cx estab/s'
   1st RX              |  'lst rx'
   CX TO               |  'cx to'
   Pattern             |  'pattern'
   Min PDU             |  'min pdu'
   Max PDU             |  'max pdu'
   Min Rate            |  'min rate'
   Max Rate            |  'max rate'
   Send Buf            |  'send buf'
   Rcv Buf             |  'rcv buf'
   CWND                |  'cwnd'
   TCP MSS             |  'tcp mss'
   Bursty              |  'bursty'
   A/B                 |  'a/b'
   Elapsed             |  'elapsed'
   Destination Addr    |  'destination addr'
   Source Addr         |  'source addr'

   Using the port_mgr_cols flag:
        '4way time (us)'
        'activity'
        'alias'
        'anqp time (us)'
        'ap'
        'beacon'
        'bps rx'
        'bps rx ll'
        'bps tx'
        'bps tx ll'
        'bytes rx ll'
        'bytes tx ll'
        'channel'
        'collisions'
        'connections'
        'crypt'
        'cx ago'
        'cx time (us)'
        'device'
        'dhcp (ms)'
        'down'
        'entity id'
        'gateway ip'
        'ip'
        'ipv6 address'
        'ipv6 gateway'
        'key/phrase'
        'login-fail'
        'login-ok'
        'logout-fail'
        'logout-ok'
        'mac'
        'mask'
```

```
        'misc'
        'mode'
        'mtu'
        'no cx (us)'
        'noise'
        'parent dev'
        'phantom'
        'port'
        'port type'
        'pps rx'
        'pps tx'
        'qlen'
        'reset'
        'retry failed'
        'rx bytes'
        'rx crc'
        'rx drop'
        'rx errors'
        'rx fifo'
        'rx frame'
        'rx length'
        'rx miss'
        'rx over'
        'rx pkts'
        'rx-rate'
        'sec'
        'signal'
        'ssid'
        'status'
        'time-stamp'
        'tx abort'
        'tx bytes'
        'tx crr'
        'tx errors'
        'tx fifo'
        'tx hb'
        'tx pkts'
        'tx wind'
        'tx-failed %'
        'tx-rate'
        'wifi retries'

    Can't decide what columns to use? You can just use 'all' to select all available columns from both tables.



optional arguments:
  -h, --help            show this help message and exit
  --help_summary        Show summary of what this script does
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path, primary use when running test in test suite
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differenciate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number / serial number of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --test_id TEST_ID     test-id for kpi.csv,  script or test name
  -o CSV_OUTFILE, --csv_outfile CSV_OUTFILE
                        --csv_outfile <Output file for csv data>
  --tty TTY             --tty "/dev/ttyUSB2" the serial interface to the AP
  --baud BAUD           --baud "9600"  AP baud rate for the serial interface
  --mgr LFMGR, --lfmgr LFMGR
                        --lfmgr <hostname for where LANforge GUI is running>
  --mgr_port LFMGR_PORT, --lfmgr_port LFMGR_PORT
                        --lfmgr_port <port LANforge GUI HTTP service is running on>
  --test_duration TEST_DURATION
                        --test_duration <how long to run>  example --time 5d (5 days) default: 3m options: number followed by d, h
  --tos TOS             --tos:  Support different ToS settings: BK | BE | VI | VO | numeric
  --debug               --debug flag present debug on  enable debugging
  -t ENDP_TYPE, --endp_type ENDP_TYPE
                        --endp_type <types of traffic> example --endp_type "lf_udp lf_tcp mc_udp"  Default: lf_udp , options: lf_u
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        --upstream_port <cross connect upstream_port> example: --upstream_port eth1
  --downstream_port DOWNSTREAM_PORT
                        --downstream_port <cross connect downstream_port> example: --downstream_port eth2
  --polling_interval POLLING_INTERVAL
                        --polling_interval <seconds>
  -r RADIO, --radio RADIO
                         --radio "radio==<number_of_wiphy stations=<=number of stations> ssid==<ssid> ssid_pw==<ssid password> sec
  --collect_layer3_data
                        --collect_layer3_data flag present creates csv files recording layer3 columns of cxs.
  --ap_read             --ap_read  flag present enable reading ap
  --ap_scheme {serial,telnet,ssh,mux_serial}
                        --ap_scheme '/dev/ttyUSB0'
  --ap_port AP_PORT     --ap_port '/dev/ttyUSB0'
  --ap_baud AP_BAUD     --ap_baud '115200'',  default='115200
  --ap_ip AP_IP         --ap_ip
  --ap_ssh_port AP_SSH_PORT
                        --ap_ssh_port
  --ap_user AP_USER     --ap_user , the user name for the ap, default = lanforge
  --ap_passwd AP_PASSWD
                        --ap_passwd, the password for the ap default = lanforge
  --ap_if_2g AP_IF_2G   --ap_if_2g eth6
  --ap_if_5g AP_IF_5G   --ap_if_5g eth7
  --ap_if_6g AP_IF_6G   --ap_if_6g eth8
  --ap_cmd_6g AP_CMD_6G
                        ap_cmd_6g 'wl -i wl2 bs_data'
  --ap_cmd_5g AP_CMD_5G
                        ap_cmd_5g 'wl -i wl1 bs_data'
  --ap_cmd_2g AP_CMD_2G
```

```
                      ap_cmd_2g 'wl -i wl0 bs_data'
  --ap_cmd_ul_6g AP_CMD_UL_6G
                      ap_cmd_ul_6g 'wl -i wl2 rx_report'
  --ap_cmd_ul_5g AP_CMD_UL_5G
                      ap_cmd_ul_5g 'wl -i wl1 rx_report'
  --ap_cmd_ul_2g AP_CMD_UL_2G
                      ap_cmd_ul_2g 'wl -i wl0 rx_report'
  --ap_chanim_cmd_6g AP_CHANIM_CMD_6G
                      ap_chanim_cmd_6g 'wl -i wl2 chanim_stats'
  --ap_chanim_cmd_5g AP_CHANIM_CMD_5G
                      ap_chanim_cmd_5g 'wl -i wl1 chanim_stats'
  --ap_chanim_cmd_2g AP_CHANIM_CMD_2G
                      ap_chanim_cmd_2g 'wl -i wl0 chanim_stats'
  --ap_scheduler_stats  --ap_scheduler_stats flag to clear stats run test then dump ul and dl stats to file
  --ap_ofdma_stats      --ap_ofdma_stats flag to clear stats run test then dumps wl -i wl1 muinfo -v and wl 0i wl0 muinof -v to fi
  --ap_test_mode        ap_test_mode flag present use ap canned data
  -amr SIDE_A_MIN_BPS, --side_a_min_bps SIDE_A_MIN_BPS
                      --side_a_min_bps, requested downstream min tx rate, comma separated list for multiple iterations.  Default
  -amp SIDE_A_MIN_PDU, --side_a_min_pdu SIDE_A_MIN_PDU
                      --side_a_min_pdu, downstream pdu size, comma separated list for multiple iterations.  Default MTU
  -bmr SIDE_B_MIN_BPS, --side_b_min_bps SIDE_B_MIN_BPS
                      --side_b_min_bps, requested upstream min tx rate, comma separated list for multiple iterations.  Default 2
  -bmp SIDE_B_MIN_PDU, --side_b_min_pdu SIDE_B_MIN_PDU
                      --side_b_min_pdu, upstream pdu size, comma separated list for multiple iterations. Default MTU
  --rates_are_totals    Treat configured rates as totals instead of using the un-modified rate for every connection.
  --multiconn MULTICONN
                      Configure multi-conn setting for endpoints.  Default is 1 (auto-helper is enabled by default as well).
  --attenuators ATTENUATORS
                      --attenuators,  comma separated list of attenuator module eids:  shelf.resource.atten-serno.atten-idx
  --atten_vals ATTEN_VALS
                      --atten_vals,  comma separated list of attenuator settings in ddb units (1/10 of db)
  --cap_ctl_out         --cap_ctl_out, switch the controller output will be captured
  --wait WAIT           --wait <time> , time to wait at the end of the test
  --sta_start_offset STA_START_OFFSET
                      Station start offset for building stations
  --no_pre_cleanup      Do not pre cleanup stations on start
  --no_cleanup          Do not cleanup before exit
  --no_stop_traffic     leave traffic running
  --use_existing_station_list
                      --use_station_list ,full eid must be given,the script will use stations from the list, no configuration on
  --existing_station_list EXISTING_STATION_LIST
                      --station_list [list of stations] , use the stations in the list , multiple station lists may be entered
  --wait_for_ip_sec WAIT_FOR_IP_SEC
                      --wait_for_ip_sec <seconds>  default : 120s
  --log_level LOG_LEVEL
                      Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                      --lf_logger_config_json <json file> , json configuration of logger

        Useful Information:
            1. Polling interval for checking traffic is fixed at 1 minute
            2. The test will generate csv file
            3. The tx/rx rates are fixed at 256000 bits per second
            4. Maximum stations per radio based on radio
```

py-scripts/test_l3_powersave_traffic.py

```
usage: test_l3_powersave_traffic.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                                    [-u UPSTREAM_PORT]
                                    [--num_stations NUM_STATIONS]
                                    [--test_id TEST_ID] [-d]
                                    [--log_level LOG_LEVEL]
                                    [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                    [--proxy [PROXY]]
                                    [--debugging DEBUGGING [DEBUGGING ...]]
                                    [--debug_log DEBUG_LOG] [--no_cleanup]
                                    [--help_summary] [--radio RADIO]
                                    [--security SECURITY] [--ssid SSID]
                                    [--passwd PASSWD]
                                    [--monitor_radio MONITOR_RADIO]

Example of creating traffic on an l3 connection


optional arguments:
  -h, --help            show this help message and exit
  --monitor_radio MONITOR_RADIO
                        --monitor_radio radio to be used in monitor creation

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
```

```
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

       test_l3_powersave_traffic.py
```

py-scripts/test_l3.py

Example report: test_l3.pdf

```
usage: test_l3.py [-h] [--local_lf_report_dir LOCAL_LF_REPORT_DIR]
                  [--results_dir_name RESULTS_DIR_NAME] [--test_rig TEST_RIG]
                  [--test_tag TEST_TAG] [--dut_hw_version DUT_HW_VERSION]
                  [--dut_sw_version DUT_SW_VERSION]
                  [--dut_model_num DUT_MODEL_NUM]
                  [--dut_serial_num DUT_SERIAL_NUM]
                  [--test_priority TEST_PRIORITY] [--test_id TEST_ID]
                  [-o CSV_OUTFILE] [--tty TTY] [--baud BAUD] [--mgr LFMGR]
                  [--mgr_port LFMGR_PORT] [--test_duration TEST_DURATION]
                  [--tos TOS] [--debug] [--log_level LOG_LEVEL]
                  [--interopt_mode] [-t ENDP_TYPE] [-u UPSTREAM_PORT]
                  [--downstream_port DOWNSTREAM_PORT]
                  [--polling_interval POLLING_INTERVAL] [-r RADIO]
                  [-amr SIDE_A_MIN_BPS] [-amp SIDE_A_MIN_PDU]
                  [-bmr SIDE_B_MIN_BPS] [-bmp SIDE_B_MIN_PDU]
                  [--rates_are_totals] [--multiconn MULTICONN]
                  [--attenuators ATTENUATORS] [--atten_vals ATTEN_VALS]
                  [--wait WAIT] [--sta_start_offset STA_START_OFFSET]
                  [--no_pre_cleanup] [--no_cleanup] [--cleanup_cx]
                  [--csv_data_to_report] [--no_stop_traffic] [--quiesce_cx]
                  [--use_existing_station_list]
                  [--existing_station_list EXISTING_STATION_LIST]
                  [--wait_for_ip_sec WAIT_FOR_IP_SEC] [--exit_on_ip_acquired]
                  [--lf_logger_config_json LF_LOGGER_CONFIG_JSON] [--ap_read]
                  [--ap_module AP_MODULE] [--ap_test_mode AP_TEST_MODE]
                  [--ap_scheme {serial,telnet,ssh,mux_serial}]
                  [--ap_serial_port AP_SERIAL_PORT]
                  [--ap_serial_baud AP_SERIAL_BAUD] [--ap_ip AP_IP]
                  [--ap_ssh_port AP_SSH_PORT]
                  [--ap_telnet_port AP_TELNET_PORT] [--ap_user AP_USER]
                  [--ap_passwd AP_PASSWD] [--ap_if_2g AP_IF_2G]
                  [--ap_if_5g AP_IF_5G] [--ap_if_6g AP_IF_6G]
                  [--ap_file AP_FILE] [--ap_band_list AP_BAND_LIST]
                  [--dowebgui DOWEBGUI] [--test_name TEST_NAME]
                  [--help_summary]

NAME: test_l3.py

PURPOSE: The Layer 3 Traffic Generation Test is designed to test the performance of the Access Point by running layer-3
         Cross-Connect Traffic.  Layer-3 Cross-Connects represent a stream of data flowing through the system under test.
         A Cross-Connect (CX) is composed of two Endpoints, each of which is associated with a particular Port (physical or virtua

         The test will create stations, create cx traffic between upstream port and stations,  run traffic. Verify
         the traffic is being transmitted and received

         * Supports creating user-specified amount stations on multiple radios
         * Supports configuring upload and download requested rates and PDU sizes.
         * Supports generating connections with different ToS values.
         * Supports generating tcp and/or UDP traffic types.
         * Supports iterating over different PDU sizes
         * Supports iterating over different requested tx rates (configurable as total or per-connection value)
         * Supports iterating over attenuation values.
         * Supports testing connection between two ethernet connection - L3 dataplane

         Generic command layout:
         -----------------------
         ./test_l3.py --mgr <ip_address> --test_duration <duration> --endp_type <traffic types> --upstream_port <port>
         --radio "radio==<radio> stations==<number stations> ssid==<ssid> ssid_pw==<ssid password>
         security==<security type: wpa2, open, wpa3>" --debug

EXAMPLE:

########################################
# Examples
########################################
Example running traffic with two radios
1. Test duration 30 minutes
2. Traffic IPv4 TCP, UDP
3. Upstream-port eth2
4. Radio #0 wiphy0 has 1 station, ssid = ssid_2g, ssid password = ssid_pw_2g  security = wpa2
5. Radio #1 wiphy1 has 2 stations, ssid = ssid_5g, ssid password = BLANK security = open
6. Create connections with TOS of BK and VI

         # The script now supports multiple radios, each specified with an individual --radio switch.

        # Interopt example Creating stations
            Interopt testing creating stations
            ./test_l3.py --lfmgr 192.168.0.103             --test_duration 60s           --polling_interval 5s            --upstr

            ./test_l3.py --lfmgr 192.168.0.103           --local_lf_report_dir /home/lanforge/html-reports/ct_id_004           -

        # Interopt using existing stations
            Interopt testing creating stations
            ./test_l3.py --lfmgr 192.168.91.50             --test_duration 60s           --polling_interval 5s            --upstr

           * UDP and TCP bi-directional test, no use of controller.
            ./test_l3.py --mgr 192.168.200.83 --endp_type 'lf_udp,lf_tcp' --upstream_port 1.1.eth1
            --radio "radio==1.1.wiphy0 stations==5 ssid==Netgear2g ssid_pw==lanforge security==wpa2"
            --radio "radio==1.1.wiphy1 stations==1 ssid==Netgear5g ssid_pw==lanforge security==wpa2"
            --test_duration 60s
```

```
   * Port resets, chooses random value between min and max
     ./test_l3.py --lfmgr 192.168.200.83 --test_duration 90s --polling_interval 10s --upstream_port eth1
     --radio 'radio==wiphy0,stations==4,ssid==Netgear2g,ssid_pw==lanforge,security==wpa2,reset_port_enable==TRUE,
     reset_port_time_min==10s,reset_port_time_max==20s' --endp_type lf_udp --rates_are_totals --side_a_min_bps=20000
     --side_b_min_bps=300000000

 # Command: (remove carriage returns)
     ./test_l3.py --lfmgr 192.168.200.83 --test_duration 30s --endp_type "lf_tcp,lf_udp" --tos "BK VI" --upstream_port 1.1
     --radio "radio==1.1.wiphy0 stations==1 ssid==Netgear2g ssid_pw==lanforge security==wpa2"

 # Have the stations continue to run after the completion of the script
     ./test_l3.py --lfmgr 192.168.200.83 --endp_type 'lf_udp,lf_tcp' --tos BK --upstream_port 1.1.eth1
     --radio 'radio==wiphy0 stations==2 ssid==Netgear2g ssid_pw==lanforge security==wpa2' --test_duration 30s
     --polling_interval 5s --side_a_min_bps 256000 --side_b_min_bps 102400000 --no_stop_traffic

 #  Have script use existing stations from previous run where traffic was not stopped and also create new stations and lea
     ./test_l3.py --lfmgr 192.168.200.83 --endp_type 'lf_udp,lf_tcp' --tos BK --upstream_port 1.1.eth1
     --radio 'radio==wiphy0 stations==2 ssid==Netgear2g ssid_pw==lanforge security==wpa2' --sta_start_offset 1000
     --test_duration 30s --polling_interval 5s --side_a_min_bps 256000 --side_b_min_bps 102400000 --use_existing_station_l
     --existing_station_list '1.1.sta0000,1.1.sta0001,1.1.sta0002' --no_stop_traffic

 # Have script use wifi_settings enable flages  ::  wifi_settings==wifi_settings,enable_flags==(ht160_enable&&wpa2_enable&
     ./test_l3.py --lfmgr 192.168.200.83 --test_duration 20s --polling_interval 5s --upstream_port 1.1.eth1
     --radio 'radio==1.1.wiphy0,stations==1,ssid==Netgear2g,ssid_pw==lanforge,security==wpa2,wifi_mode==0,wifi_settings==w
     --radio 'radio==1.1.wiphy1,stations==1,ssid==Netgear5g,ssid_pw==lanforge,security==wpa2,wifi_mode==0,wifi_settings==w
     --radio 'radio==1.1.wiphy2,stations==1,ssid==Netgear2g,ssid_pw==lanforge,security==wpa2,wifi_mode==0,wifi_settings==w
     --endp_type lf_udp --rates_are_totals --side_a_min_bps=20000 --side_b_min_bps=300000000 --test_rig ID_003 --test_tag
     --dut_hw_version 1.0 --dut_serial_num 12345678 --log_level debug

 # Setting wifi_settings per radio
    ./test_l3.py
    --lfmgr 192.168.100.116
    --local_lf_report_dir /home/lanforge/html-reports/
    --test_duration 15s
    --polling_interval 5s
    --upstream_port eth2
    --radio "radio==wiphy1 stations==4 ssid==asus11ax-5 ssid_pw==hello123 security==wpa2  mode==0 wifi_settings==wifi_sett
    --endp_type lf_udp
    --rates_are_totals
    --side_a_min_bps=20000
    --side_b_min_bps=300000000
    --test_rig CT-US-001
    --test_tag 'test_l3'

 # Example : LAN-1927  WPA2-TLS-Configuration
    ./test_l3.py
     --lfmgr 192.168.0.103
     --test_duration 20s
     --polling_interval 5s
     --upstream_port 1.1.eth2
     --radio 'radio==wiphy1,stations==1,ssid==ax88u_5g,ssid_pw==[BLANK],security==wpa2,wifi_settings==wifi_settings,wifi_m
     --endp_type lf_udp
     --rates_are_totals
     --side_a_min_bps=256000
     --side_b_min_bps=300000000
     --test_rig ID_003
     --test_tag 'test_l3'
     --dut_model_num GT-AXE11000
     --dut_sw_version 3.0.0.4.386_44266
     --dut_hw_version 1.0
     --dut_serial_num 12345678
     --log_level debug

# Example : LAN-1927  WPA2-TTLS-Configuration
    ./test_l3.py
     --lfmgr 192.168.0.103
     --test_duration 20s
     --polling_interval 5s
     --upstream_port 1.1.eth2
     --radio 'radio==wiphy1,stations==1,ssid==ax88u_5g,ssid_pw==[BLANK],security==wpa2,wifi_settings==wifi_settings,wifi_m
     --endp_type lf_udp
     --rates_are_totals
     --side_a_min_bps=256000
     --side_b_min_bps=300000000
     --test_rig ID_003
     --test_tag 'test_l3'
     --dut_model_num GT-AXE11000
     --dut_sw_version 3.0.0.4.386_44266
     --dut_hw_version 1.0
     --dut_serial_num 12345678
     --log_level debug

# Example : LAN-1927  WPA3-TTLS-Configuration
    ./test_l3.py
     --lfmgr 192.168.0.103
     --test_duration 20s
     --polling_interval 5s
     --upstream_port 1.1.eth2
     --radio 'radio==wiphy1,stations==1,ssid==ax88u_5g,ssid_pw==[BLANK],security==wpa3,wifi_settings==wifi_settings,wifi_m
     --endp_type lf_ud
     --rates_are_totals
     --side_a_min_bps=256000
     --side_b_min_bps=300000000
     --test_rig ID_003
     --test_tag 'test_l3'
     --dut_model_num GT-AXE11000
     --dut_sw_version 3.0.0.4.386_44266
     --dut_hw_version 1.0
     --dut_serial_num 12345678
     --log_level debug

# Example : LAN-1927  WPA3-TLS-Configuration
    ./test_l3.py
     --lfmgr 192.168.0.103
     --test_duration 20s
     --polling_interval 5s
```

```
            --upstream_port 1.1.eth2
            --radio 'radio==wiphy1,stations==1,ssid==ax88u_5g,ssid_pw==[BLANK],security==wpa3,wifi_settings==wifi_settings,wifi_m
            --endp_type lf_udp
            --rates_are_totals
            --side_a_min_bps=256000
            --side_b_min_bps=300000000
            --test_rig ID_003
            --test_tag 'test_l3'
            --dut_model_num GT-AXE11000
            --dut_sw_version 3.0.0.4.386_44266
            --dut_hw_version 1.0
            --dut_serial_num 12345678
            --log_level debug

SCRIPT_CLASSIFICATION:  Creation & Runs Traffic

SCRIPT_CATEGORIES:  Performance, Functional,  KPI Generation,  Report Generation

NOTES:

################################
# Command switches
################################

--mgr <hostname for where LANforge GUI is running>',default='localhost'
-d  / --test_duration <how long to run>  example --time 5d (5 days) default: 3m options: number followed by d, h, m or s',default=
--tos:  Support different ToS settings: BK | BE | VI | VO | numeric',default="BE"
--debug:  Enable debugging',default=False
-t  / --endp_type <types of traffic> example --endp_type "lf_udp lf_tcp mc_udp"  Default: lf_udp , options: lf_udp, lf_udp6, lf_tc
                    default='lf_udp', type=valid_endp_types
-u / --upstream_port <cross connect upstream_port> example: --upstream_port eth1',default='eth1')
-o / --outfile <Output file for csv data>", default='longevity_results'

<duration>: number followed by one of the following
d - days
h - hours
m - minutes
s - seconds

<traffic type>:
lf_udp  : IPv4 UDP traffic
lf_tcp  : IPv4 TCP traffic
lf_udp6 : IPv6 UDP traffic
lf_tcp6 : IPv6 TCP traffic
mc_udp  : IPv4 multi cast UDP traffic
mc_udp6 : IPv6 multi cast UDP traffic

<tos>:
BK, BE, VI, VO:  Optional wifi related Tos Settings.  Or, use your preferred numeric values. Cross connects type of service

    * Data 0 (Best Effort, BE): Medium priority queue, medium throughput and delay.
            Most traditional IP data is sent to this queue.
    * Data 1 (Background, BK): Lowest priority queue, high throughput. Bulk data that requires maximum throughput and
            is not time-sensitive is sent to this queue (FTP data, for example).
    * Data 2 (Video, VI): High priority queue, minimum delay. Time-sensitive data such as Video and other streaming
            media are automatically sent to this queue.
    * Data 3 (Voice, VO): Highest priority queue, minimum delay. Time-sensitive data such as Voice over IP (VoIP)
            is automatically sent to this Queue.

<wifi_mode>:
    Input       : Enum Val  : Shown by nc_show_ports

    AUTO        |  0        #  802.11
    802.11a     |  1        #  802.11a
    b           |  2        #  802.11b
    g           |  3        #  802.11g
    abg         |  4        #  802.11abg
    abgn        |  5        #  802.11abgn
    bgn         |  6        #  802.11bgn
    bg          |  7        #  802.11bg
    abgnAC      |  8        #  802.11abgn-AC
    anAC        |  9        #  802.11an-AC
    an          | 10        #  802.11an
    bgnAC       | 11        #  802.11bgn-AC
    abgnAX      | 12        #  802.11abgn-AX
                            #     a/b/g/n/AC/AX (dual-band AX) support
    bgnAX       | 13        #  802.11bgn-AX
    anAX        | 14        #  802.11an-AX
    aAX         | 15        #  802.11a-AX (6E disables /n and /ac)

wifi_settings flags are currently defined as:
    wpa_enable          | 0x10         # Enable WPA
    # Use Custom wpa_supplicant config file.
    custom_conf         | 0x20
    # Use wpa_supplicant configured for WEP encryption.
    wep_enable          | 0x200
    # Use wpa_supplicant configured for WPA2 encryption.
    wpa2_enable         | 0x400
    # Disable HT-40 even if hardware and AP support it.
    ht40_disable        | 0x800
    # Enable SCAN-SSID flag in wpa_supplicant.
    scan_ssid           | 0x1000
    # Use passive scanning (don't send probe requests).
    passive_scan        | 0x2000
    disable_sgi         | 0x4000       # Disable SGI (Short Guard Interval).
    # OK-To-Migrate (Allow station migration between LANforge radios)
    lf_sta_migrate      | 0x8000
    # Verbose-Debug:  Increase debug info in wpa-supplicant and hostapd logs.
    verbose             | 0x10000
    # Enable 802.11u (Interworking) feature.
    8021lu_enable       | 0x20000
    # Enable 802.11u (Interworking) Auto-internetworking feature.  Always enabled currently.
    8021lu_auto         | 0x40000
    # AP Provides access to internet (802.11u Interworking)
    8021lu_gw           | 0x80000
    # AP requires additional step for access (802.11u Interworking)
    8021lu_additional   | 0x100000
```

```
    # AP claims emergency services reachable (802.11u Interworking)
    80211u_e911           | 0x200000
    # AP provides Unauthenticated emergency services (802.11u Interworking)
    80211u_e911_unauth    | 0x400000
    # Enable Hotspot 2.0 (HS20) feature.  Requires WPA-2.
    hs20_enable           | 0x800000
    # AP:  Disable DGAF (used by HotSpot 2.0).
    disable_gdaf          | 0x1000000
    8021x_radius          | 0x2000000    # Use 802.1x (RADIUS for AP).
    # Enable oportunistic PMSKA caching for WPA2 (Related to 802.11r).
    80211r_pmska_cache    | 0x4000000
    # Disable HT80 (for AC chipset NICs only)
    disable_ht80          | 0x8000000
    ibss_mode             | 0x20000000   # Station should be in IBSS mode.
    # Enable OSEN protocol (OSU Server-only Authentication)
    osen_enable           | 0x40000000
    # Disable automatic station roaming based on scan results.
    disable_roam          | 0x80000000
    ht160_enable          | 0x100000000  # Enable HT160 mode.
    # Disable fast_reauth option for virtual stations.
    disable_fast_reauth   | 0x200000000
    mesh_mode             | 0x400000000  # Station should be in MESH mode.
    # Station should enable power-save.  May not work in all drivers/configurations.
    power_save_enable     | 0x800000000
    create_admin_down     | 0x1000000000 # Station should be created admin-down.
    # WDS station (sort of like a lame mesh), not supported on ath10k
    wds-mode              | 0x2000000000
    # Do not include supported-oper-class-IE in assoc requests.  May work around AP bugs.
    no-supp-op-class-ie   | 0x4000000000
    # Enable/disable tx-offloads, typically managed by set_wifi_txo command
    txo-enable            | 0x8000000000
    use-wpa3              | 0x10000000000 # Enable WPA-3 (SAE Personal) mode.
    use-bss-transition    | 0x80000000000 # Enable BSS transition.
    disable-twt           | 0x100000000000 # Disable TWT mode

For wifi_extra_keys syntax :
    telnet <lanforge ip> 4001
    type: help set_wifi_extra
wifi_extra keys:
                key_mgmt  (Key Mangement)
                pairwise  (Pairwise Ciphers)
                group   (Group Ciphers)
                psk     (WPA PSK)
                wep_key
                ca_cert (CA Cert File)
                eap     (EAP Methods) EAP method: MD5, MSCHAPV2, OTP, GTC, TLS, PEAP, TTLS. (note different the GUI no appended EA
                identity    (EAP Identity)
                anonymous_identity  (EAP Anon Identity)
                phase1  (Phase-1)
                phase2  (Phase-2)
                passwd  (EAP Password)
                pin (EAP Pin)
                pac_file    (PAC file)
                private_key (Private Key)
                pk_password (PK Password)
                hessid="00:00:00:00:00:00"
                realm   (Realm)
                client_cert (Client Cert)
                imsi    (IMSI)
                milenage    (Milenage)
                domain  (Domain)
                roaming_consortium  (Consortium)
                venue_group ()
                network_type    (Network Auth)
                ipaddr_type_avail   ()
                network_auth_type ()
                anqp_3gpp_cell_net ()

                ieee80211w :   0,1,2

Multicast traffic :
        Multicast traffic default IGMP Address in the range of 224.0.0.0 to 239.255.255.255,
        so I have provided 224.9.9.9 as IGMP address and IGMP Dest port as 9999 and MIN-IP PORT as 9999.
        these values must be same on the eth1(server side) and client side, then the traffic will run.

================================================================================
 ** FURTHER INFORMATION **
    Using the layer3_cols flag:

    Currently the output function does not support inputting the columns in layer3_cols the way they are displayed in the GUI. Thi
    certain columns in the GUI in your final report, please match the according GUI column display to it's counterpart to have the
    your report.

    GUI Column Display       Layer3_cols argument to type in (to print in report)

    Name                |  'name'
    EID                 |  'eid'
    Run                 |  'run'
    Mng                 |  'mng'
    Script              |  'script'
    Tx Rate             |  'tx rate'
    Tx Rate (1 min)     |  'tx rate (1 min)'
    Tx Rate (last)      |  'tx rate (last)'
    Tx Rate LL          |  'tx rate ll'
    Rx Rate             |  'rx rate'
    Rx Rate (1 min)     |  'rx rate (1 min)'
    Rx Rate (last)      |  'rx rate (last)'
    Rx Rate LL          |  'rx rate ll'
    Rx Drop %           |  'rx drop %'
    Tx PDUs             |  'tx pdus'
    Tx Pkts LL          |  'tx pkts ll'
    PDU/s TX            |  'pdu/s tx'
    Pps TX LL           |  'pps tx ll'
    Rx PDUs             |  'rx pdus'
    Rx Pkts LL          |  'pps rx ll'
    PDU/s RX            |  'pdu/s tx'
    Pps RX LL           |  'pps rx ll'
```

```
Delay               |  'delay'
Dropped             |  'dropped'
Jitter              |  'jitter'
Tx Bytes            |  'tx bytes'
Rx Bytes            |  'rx bytes'
Replays             |  'replays'
TCP Rtx             |  'tcp rtx'
Dup Pkts            |  'dup pkts'
Rx Dup %            |  'rx dup %'
OOO Pkts            |  'ooo pkts'
Rx OOO %            |  'rx ooo %'
RX Wrong Dev        |  'rx wrong dev'
CRC Fail            |  'crc fail'
RX BER              |  'rx ber'
CX Active           |  'cx active'
CX Estab/s          |  'cx estab/s'
1st RX              |  '1st rx'
CX TO               |  'cx to'
Pattern             |  'pattern'
Min PDU             |  'min pdu'
Max PDU             |  'max pdu'
Min Rate            |  'min rate'
Max Rate            |  'max rate'
Send Buf            |  'send buf'
Rcv Buf             |  'rcv buf'
CWND                |  'cwnd'
TCP MSS             |  'tcp mss'
Bursty              |  'bursty'
A/B                 |  'a/b'
Elapsed             |  'elapsed'
Destination Addr    |  'destination addr'
Source Addr         |  'source addr'

Using the port_mgr_cols flag:
     '4way time (us)'
     'activity'
     'alias'
     'anqp time (us)'
     'ap'
     'beacon'
     'bps rx'
     'bps rx ll'
     'bps tx'
     'bps tx ll'
     'bytes rx ll'
     'bytes tx ll'
     'channel'
     'collisions'
     'connections'
     'crypt'
     'cx ago'
     'cx time (us)'
     'device'
     'dhcp (ms)'
     'down'
     'entity id'
     'gateway ip'
     'ip'
     'ipv6 address'
     'ipv6 gateway'
     'key/phrase'
     'login-fail'
     'login-ok'
     'logout-fail'
     'logout-ok'
     'mac'
     'mask'
     'misc'
     'mode'
     'mtu'
     'no cx (us)'
     'noise'
     'parent dev'
     'phantom'
     'port'
     'port type'
     'pps rx'
     'pps tx'
     'qlen'
     'reset'
     'retry failed'
     'rx bytes'
     'rx crc'
     'rx drop'
     'rx errors'
     'rx fifo'
     'rx frame'
     'rx length'
     'rx miss'
     'rx over'
     'rx pkts'
     'rx-rate'
     'sec'
     'signal'
     'ssid'
     'status'
     'time-stamp'
     'tx abort'
     'tx bytes'
     'tx crr'
     'tx errors'
     'tx fifo'
     'tx hb'
     'tx pkts'
     'tx wind'
     'tx-failed %'
     'tx-rate'
```

```
        'wifi retries'

    Can't decide what columns to use? You can just use 'all' to select all available columns from both tables.

STATUS: Functional

VERIFIED_ON:   18-JULY-2023,
             GUI Version:  5.4.6
             Kernel Version: 5.19.17+

LICENSE:
          Free to distribute and modify. LANforge systems must be licensed.
          Copyright 2023 Candela Technologies Inc

INCLUDE_IN_README: False



optional arguments:
  -h, --help            show this help message and exit
  --help_summary        Show summary of what this script does

arguments defined in test_l3.py file:
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path (lanforge/html-reports), primary used when making another d
  --results_dir_name RESULTS_DIR_NAME
                        the name of the directory that contains the output from the test /lanforge/html-reports/<results_dir_name>
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differenciate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number / serial number of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --test_id TEST_ID     test-id for kpi.csv,  script or test name
  -o CSV_OUTFILE, --csv_outfile CSV_OUTFILE
                        --csv_outfile <Output file for csv data>
  --tty TTY             --tty "/dev/ttyUSB2" the serial interface to the AP
  --baud BAUD           --baud "9600"  AP baud rate for the serial interface
  --mgr LFMGR, --lfmgr LFMGR
                        --lfmgr <hostname for where LANforge GUI is running>
  --mgr_port LFMGR_PORT, --lfmgr_port LFMGR_PORT
                        --lfmgr_port <port LANforge GUI HTTP service is running on>
  --test_duration TEST_DURATION
                        --test_duration <how long to run>  example --time 5d (5 days) default: 3m options: number followed by d, h
  --tos TOS             --tos:  Support different ToS settings: BK,BE,VI,VO,numeric
  --debug               --debug this will enable debugging in py-json method
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --interopt_mode       For Interopt continue to try running even if some clients do not get an IP.
  -t ENDP_TYPE, --endp_type ENDP_TYPE
                        --endp_type <types of traffic> example --endp_type "lf_udp lf_tcp mc_udp"  Default: lf_udp , options: lf_u
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        --upstream_port <cross connect upstream_port> example: --upstream_port eth1
  --downstream_port DOWNSTREAM_PORT
                        --downstream_port <cross connect downstream_port>  for use when downstream is ethernet (eth to eth connect
  --polling_interval POLLING_INTERVAL
                        --polling_interval <seconds>
  -r RADIO, --radio RADIO
                         --radio "radio==<number_of_wiphy stations==<number of stations> ssid==<ssid> ssid_pw==<ssid password> sec
  -amr SIDE_A_MIN_BPS, --side_a_min_bps SIDE_A_MIN_BPS, --upload_min_bps SIDE_A_MIN_BPS
                        --side_a_min_bps, requested downstream min tx rate at stations / client, comma separated list for multiple
                                    When running with tcp/udp traffic along with mcast , mcast will ignore the upload value
  -amp SIDE_A_MIN_PDU, --side_a_min_pdu SIDE_A_MIN_PDU
                        --side_a_min_pdu, downstream pdu size, comma separated list for multiple iterations.  Default MTU
  -bmr SIDE_B_MIN_BPS, --download_min_bps SIDE_B_MIN_BPS, --side_b_min_bps SIDE_B_MIN_BPS, --do SIDE_B_MIN_BPS
                        --side_b_min_bps or --download_min_bps, requested upstream min tx rate, comma separated list for multiple
                                    When runnign with tcp/udp and mcast will use this value
  -bmp SIDE_B_MIN_PDU, --side_b_min_pdu SIDE_B_MIN_PDU
                        --side_b_min_pdu, upstream pdu size, comma separated list for multiple iterations. Default MTU
  --rates_are_totals    Treat configured rates as totals instead of using the un-modified rate for every connection.
  --multiconn MULTICONN
                        Configure multi-conn setting for endpoints.  Default is 1 (auto-helper is enabled by default as well).
  --attenuators ATTENUATORS
                        --attenuators,  comma separated list of attenuator module eids:  shelf.resource.atten-serno.atten-idx
  --atten_vals ATTEN_VALS
                        --atten_vals,  comma separated list of attenuator settings in ddb units (1/10 of db)
  --wait WAIT           --wait <time> , time to wait at the end of the test
  --sta_start_offset STA_START_OFFSET
                        Station start offset for building stations
  --no_pre_cleanup      Do not pre cleanup stations on start
  --no_cleanup          Do not cleanup before exit
  --cleanup_cx          cleanup cx before exit
  --csv_data_to_report  collected interval data in csv for each cx will be put in report
  --no_stop_traffic     leave traffic running
  --quiesce_cx          --quiesce store true,  allow the cx to drain then stop so as to not have rx drop pkts
  --use_existing_station_list
                        --use_station_list ,full eid must be given,the script will use stations from the list, no configuration on
  --existing_station_list EXISTING_STATION_LIST
                        --station_list [list of stations] , use the stations in the list , multiple station lists may be entered
  --wait_for_ip_sec WAIT_FOR_IP_SEC
                        --wait_for_ip_sec <seconds>  default : 120s
  --exit_on_ip_acquired
                        --exit_on_ip_acquired store true
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --ap_read             --ap_read  flag present enable reading ap
  --ap_module AP_MODULE
                        series module
  --ap_test_mode AP_TEST_MODE
                        --ap_mode
  --ap_scheme {serial,telnet,ssh,mux_serial}
```

```
                      --ap_scheme '/dev/ttyUSB0'
  --ap_serial_port AP_SERIAL_PORT
                      --ap_serial_port '/dev/ttyUSB0'
  --ap_serial_baud AP_SERIAL_BAUD
                      --ap_baud '115200'',  default='115200
  --ap_ip AP_IP       --ap_ip
  --ap_ssh_port AP_SSH_PORT
                      --ap_ssh_port
  --ap_telnet_port AP_TELNET_PORT
                      --ap_telnet_port
  --ap_user AP_USER   --ap_user , the user name for the ap, default = lanforge
  --ap_passwd AP_PASSWD
                      --ap_passwd, the password for the ap default = lanforge
  --ap_if_2g AP_IF_2G   --ap_if_2g eth6
  --ap_if_5g AP_IF_5G   --ap_if_5g eth7
  --ap_if_6g AP_IF_6G   --ap_if_6g eth8
  --ap_file AP_FILE     --ap_file 'ap_file.txt'
  --ap_band_list AP_BAND_LIST
                      --ap_band_list '2g,5g,6g' supported bands
  --dowebgui DOWEBGUI   --dowebgui True  if running through webgui
  --test_name TEST_NAME
                      Test name when running through webgui

        Useful Information:
            1. Polling interval for checking traffic is fixed at 1 minute
            2. The test will generate csv file
            3. The tx/rx rates are fixed at 256000 bits per second
            4. Maximum stations per radio based on radio
```

py-scripts/test_l3_scenario_throughput.py

```
usage: test_l3_scenario_throughput.py [-h] [-m MANAGER] [-sc SCENARIO]
                                      [-t DURATION] [-o REPORT_NAME]
                                      [-td TEST_DETAIL] [--help_summary]

Test Scenario of DUT Temperature measurement along with simultaneous throughput on VAP as well as stations

optional arguments:
  -h, --help            show this help message and exit
  -m MANAGER, --manager MANAGER
                        Enter the address of Lanforge Manager (By default localhost)
  -sc SCENARIO, --scenario SCENARIO
                        Enter the Name of the Scenario you want to load (by Default DFLT)
  -t DURATION, --duration DURATION
                        Enter the Time for which you want to run test
  -o REPORT_NAME, --report_name REPORT_NAME
                        Enter the Name of the Output file ('Report.xlsx')
  -td TEST_DETAIL, --test_detail TEST_DETAIL
                        Enter the Test Detail in Quotes
  --help_summary        Show summary of what this script does
```

py-scripts/test_l3_unicast_traffic_gen.py

```
usage: test_l3_unicast_traffic_gen.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                                      [-u UPSTREAM_PORT]
                                      [--num_stations NUM_STATIONS]
                                      [--test_id TEST_ID] [-d]
                                      [--log_level LOG_LEVEL]
                                      [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                                      [--proxy [PROXY]]
                                      [--debugging DEBUGGING [DEBUGGING ...]]
                                      [--debug_log DEBUG_LOG] [--no_cleanup]
                                      [--help_summary] [--radio RADIO]
                                      [--security SECURITY] [--ssid SSID]
                                      [--passwd PASSWD]
                                      [--test_duration TEST_DURATION]
                                      [-t ENDP_TYPE]
                                      [--radio_list [RADIO_LIST [RADIO_LIST ...]]]

test_l3_unicast_traffic_gen.py:
--------------------
Basic Idea:

create stations, create traffic between upstream port and stations,  run traffic.
The traffic on the stations will be checked once per minute to verify that traffic is transmitted
and received.

Test will exit on failure of not receiving traffic for one minute on any station.

Scripts are executed from: ./lanforge/py-scripts

Stations start counting form zero,  thus stations count from zero - number of las

Generic command layout:
python ./test_l3_unicast_traffic_gen.py
        --test_duration <duration>
        --endp_type <traffic type>
        --upstream_port <port>
        --radio <radio_name> <num_stations> <ssid> <ssid_password>

Note:
multiple --radio switches may be entered up to the number of radios available:
--radio <radio 0> <number of stations> <ssid> <ssid password>  --radio <radio 01> <number of stations> <ssid> <ssid password>

<duration>: number followed by one of the following
            d - days
            h - hours
            m - minutes
            s - seconds

<traffic type>:
            lf_udp  : IPv4 UDP traffic
            lf_tcp  : IPv4 TCP traffic
            lf_udp6 : IPv6 UDP traffic
            lf_tcp6 : IPv6 TCP traffic
```

```
Example:
    1. Test duration 4 minutes
    2. Traffic IPv4 TCP
    3. Upstream-port eth1
    4. Radio #1 wiphy0 has 32 stations, ssid = candelaTech-wpa2-x2048-4-1, ssid password = candelaTech-wpa2-x2048-4-1
    5. Radio #2 wiphy1 has 64 stations, ssid = candelaTech-wpa2-x2048-5-3, ssid password = candelaTech-wpa2-x2048-5-3

Example:
Some of the command line switches are in /py-json/lfcli_base.py
python3 .\test_l3_unicast_traffic_gen.py --lfmgr --test_duration 4m --endp_type lf_tcp --upstream_port eth1



optional arguments:
  -h, --help            show this help message and exit
  --test_duration TEST_DURATION
                        --test_duration <how long to run>  example --time 5d (5 days) default: 3m options: number followed by d, h
  -t ENDP_TYPE, --endp_type ENDP_TYPE
                        --endp_type <type of traffic> example --endp_type lf_udp, default: lf_udp , options: lf_udp, lf_udp6, lf_t

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

required arguments:
  --radio_list [RADIO_LIST [RADIO_LIST ...]]

                                                    --radio_list  <number_of_wiphy> <number of last station> <ssid> <ssid pass
                                                    if radio list is 4 in length the security defaults to wpa2,  all other par

                                                    if radio list is 5 in length, security is set, all other parameters are 'D

                                                    if radio list is 6 in length, key management is set to OWE, security needs
                                                        key_management = OWE

                                                    if radio list is 9 then all parameters are set
                                                        Key_management = WPA-EAP-SHA256
                                                        pairwise_ciphers = CCMP-256
                                                        group_cipher = 'GCMP-256'
                                                        eap_methods = EAP-PEAP

                                                    (WPA2 TLS) [14]
                                                    if radio list is
                                                        Key_management = WPA-EAP
                                                        EAP Methods = EAP-TLS
                                                        EAP Identity = testuser
                                                        EAP Password = testpasswd
                                                        Pirvate Key = /home/lanforge/client.p12
                                                        CA Cert File = /home/lanforge/ca.pem
                                                        PK Passwrod = lanforge
                                                        Ieee80211w  = Disabled (0)
                                                        Advanced/8021x = checked
                                                        Enabled PKC = checked

                                                    (WPA2 TTLS) [10]
                                                    if radio list is
                                                        Key_management = WPA-EAP
                                                        EAP Methods = EAP-TTLS
                                                        EAP Identity = testuser
                                                        EAP Password = testpasswd
                                                        PK Password = lanforge
                                                        Advanced/8021x = checked


                                                    (WPA3 TLS) [16]
                                                    if radio list is
                                                        Key_management = WPA-EAP-SUITE-B-192
                                                        Pairwise Ciphers = GCMP-256 (wpa3)
                                                        Group Ciphers = GCMP - 256 (wpa3)
                                                        EAP Methods = EAP-TLS
                                                        EAP Identity = testuser
                                                        EAP Password = testpasswd
                                                        Pirvate Key = /home/lanforge/client.p12
                                                        CA Cert File = /home/lanforge/ca.pem
```

```
                                                    PK Passwrod = lanforge
                                                    Ieee80211w  = Required (2)
                                                    Advanced/8021x = checked
                                                    Enabled PKC = checked


                                                (WPA3 TTLS) [12]
                                                if radio list is
                                                    Key_management = WPA-EAP
                                                    Pairwise Ciphers = GCMP-256 (wpa3)
                                                    Group Ciphers = GCMP-256 (wpa3)
                                                    EAP Methods = EAP-TLS
                                                    EAP Identity = testuser
                                                    EAP Password = testpasswd
                                                    Ieee80211w  = Required (2)
                                                    Advanced/8021x = checked

                                                    example:
                                                --radio_list wiphy0 1 axe11000_6g lf_axe11000_6g wpa3 WPA-EAP-SHA256 CCMP-


                                                Acceptable lengths are 4,5,6,9, 10 (wpa2 TTLS), 12 (wpa3 TTLS), 14 (wpa2 T


Useful Information:

1. Polling interval for checking traffic is fixed at 1 minute
2. The test will exit when traffic has not changed on a station for 1 minute
3. The tx/rx rates are fixed at 256000 bits per second
4. Maximum stations per radio is 64

5. radio list order  --radio_list  <number_of_wiphy> <number of last station> <ssid> <ssid password> <security> <key_management> <
```

py-scripts/test_l3_WAN_LAN.py

```
usage: test_l3_WAN_LAN.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                          [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                          [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                          [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                          [--proxy [PROXY]]
                          [--debugging DEBUGGING [DEBUGGING ...]]
                          [--debug_log DEBUG_LOG] [--no_cleanup]
                          [--help_summary] [--radio RADIO]
                          [--security SECURITY] [--ssid SSID]
                          [--passwd PASSWD] [--a_min A_MIN] [--b_min B_MIN]
                          [--test_duration TEST_DURATION]
                          [--upstream_subnets UPSTREAM_SUBNETS]
                          [--upstream_nexthop UPSTREAM_NEXTHOP]
                          [--local_subnets LOCAL_SUBNETS]
                          [--local_nexthop LOCAL_NEXTHOP] [--rdd_ip RDD_IP]
                          [--rdd_gateway RDD_GATEWAY]
                          [--rdd_netmask RDD_NETMASK] [--vr_name VR_NAME]

test_l3_WAN_LAN.py:
--------------------
This script is for testing WAN to LAN traffic.


optional arguments:
  -h, --help            show this help message and exit
  --a_min A_MIN         --a_min bps rate minimum for side_a
  --b_min B_MIN         --b_min bps rate minimum for side_b
  --test_duration TEST_DURATION
                        --test_duration sets the duration of the test
  --upstream_subnets UPSTREAM_SUBNETS
                        --upstream_subnets sets the subnets used by the upstream vrcx
  --upstream_nexthop UPSTREAM_NEXTHOP
                        --upstream_nexthop sets the nexthop used by the upstream vrcx, should be rdd gateway
  --local_subnets LOCAL_SUBNETS
                        --local_subnets sets the subnets used by the rdd vrcx
  --local_nexthop LOCAL_NEXTHOP
                        --local_nexthop sets the nexthop used by the upstream vrcx, should be upstream ip
  --rdd_ip RDD_IP       --rdd_ip sets the ip to be used by the rdd
  --rdd_gateway RDD_GATEWAY
                        --rdd_gateway sets the gateway to be used by the rdd
  --rdd_netmask RDD_NETMASK
                        --rdd_netmask sets the netmask to be used by the rdd
  --vr_name VR_NAME     --vr_name sets the name to be used by the virtual router

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
```

```
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

        Useful Information:
            1. TBD
```

py-scripts/test_l4.py

```
usage: test_l4 [-h] [--mgr MGR] [--mgr_port MGR_PORT] [-u UPSTREAM_PORT]
               [--num_stations NUM_STATIONS] [--test_id TEST_ID] [-d]
               [--log_level LOG_LEVEL]
               [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
               [--proxy [PROXY]] [--debugging DEBUGGING [DEBUGGING ...]]
               [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
               [--radio RADIO] [--security SECURITY] [--ssid SSID]
               [--passwd PASSWD] [--requests_per_ten REQUESTS_PER_TEN]
               [--rpt_timer RPT_TIMER] [--num_tests NUM_TESTS] [--url URL]
               [--test_duration TEST_DURATION]
               [--target_per_ten TARGET_PER_TEN] [--mode MODE] [--ap AP]
               [--report_file REPORT_FILE] [--output_format OUTPUT_FORMAT]
               [--ftp] [--test_type TEST_TYPE] [--ftp_user FTP_USER]
               [--ftp_passwd FTP_PASSWD] [--dest DEST] [--source SOURCE]
               [--local_lf_report_dir LOCAL_LF_REPORT_DIR] [--lf_user LF_USER]
               [--lf_passwd LF_PASSWD] [--test_rig TEST_RIG]
               [--test_tag TEST_TAG] [--dut_hw_version DUT_HW_VERSION]
               [--dut_sw_version DUT_SW_VERSION]
               [--dut_model_num DUT_MODEL_NUM]
               [--dut_serial_num DUT_SERIAL_NUM]
               [--test_priority TEST_PRIORITY] [--csv_outfile CSV_OUTFILE]
               [--l4_7_port_name L4_7_PORT_NAME]

----------------------------
Layer-4 Test Script - test_l4.py
----------------------------
Summary:
This script will create stations and endpoints to generate and verify layer-4 traffic by monitoring the urls/s,
 bytes-rd, or bytes-wr attribute of the endpoints.
----------------------------
Generic command example:
./test_l4.py --mgr <ip_address> --upstream_port eth1 --radio wiphy0 --num_stations 3 --security wpa2
--ssid <ssid> --passwd <password> --test_duration 2m --url "ul http://upstream_port_ip /dev/null"
--requests_per_ten 600 --test_type bytes-wr --debug
----------------------------



optional arguments:
  -h, --help            show this help message and exit

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

arguments defined in test_l4.py file:
  --requests_per_ten REQUESTS_PER_TEN
                        --requests_per_ten number of request per ten minutes
  --rpt_timer RPT_TIMER
                        --rpt_timer report timer. Enter in milliseconds
  --num_tests NUM_TESTS
                        --num_tests number of tests to run. Each test runs 10 minutes
  --url URL
                                                --url specifies upload/download, IP of upstream eth port connected to Access Point
                                                /dev/null to discard the data example:
                                                Example 'dl http://upstream_port_ip /dev/null'  if the upsteam_port_ip is the stri
                                                'upstream_port_ip' then the upstream port ip will be read at run time
```

```
  --test_duration TEST_DURATION
                        duration of test
  --target_per_ten TARGET_PER_TEN
                        --target_per_ten target number of request per ten minutes. test will check for 90 percent this value
  --mode MODE           Used to force mode of stations
  --ap AP               Used to force a connection to a particular AP
  --report_file REPORT_FILE
                        where you want to store monitor results in output_format
  --output_format OUTPUT_FORMAT
                        'csv', 'json', 'html', 'stata', 'pickle', 'xlsx'
  --ftp                 Use ftp for the test
  --test_type TEST_TYPE
                        Choose type of test to run {urls, bytes-rd, bytes-wr}
  --ftp_user FTP_USER   --ftp_user sets the username to be used for ftp
  --ftp_passwd FTP_PASSWD
                        --ftp_user sets the password to be used for ftp
  --dest DEST           --dest specifies the destination for the file, should be used when downloading
  --source SOURCE       --source specifies the source of the file, should be used when uploading
  --local_lf_report_dir LOCAL_LF_REPORT_DIR
                        --local_lf_report_dir override the report path, primary use when running test in test suite
  --lf_user LF_USER     --lf_user lanforge user name
  --lf_passwd LF_PASSWD
                        --lf_passwd lanforge password
  --test_rig TEST_RIG   test rig for kpi.csv, testbed that the tests are run on
  --test_tag TEST_TAG   test tag for kpi.csv,  test specific information to differenciate the test
  --dut_hw_version DUT_HW_VERSION
                        dut hw version for kpi.csv, hardware version of the device under test
  --dut_sw_version DUT_SW_VERSION
                        dut sw version for kpi.csv, software version of the device under test
  --dut_model_num DUT_MODEL_NUM
                        dut model for kpi.csv,  model number / name of the device under test
  --dut_serial_num DUT_SERIAL_NUM
                        dut serial for kpi.csv, serial number / serial number of the device under test
  --test_priority TEST_PRIORITY
                        dut model for kpi.csv,  test-priority is arbitrary number
  --csv_outfile CSV_OUTFILE
                        --csv_outfile <prepend input to generated file for csv data>
  --l4_7_port_name L4_7_PORT_NAME, --downstream_port L4_7_PORT_NAME
                        --l4_7_port_name to select port to run L4_7 traffic

              This script will monitor the urls/s, bytes-rd, or bytes-wr attribute of the endpoints.
```

py-scripts/test_status_msg.py

```
usage: ./test_status_msg.py [-h] [--mgr MGR] [--mgr_port MGR_PORT] [--debug]
                            [--log_level LOG_LEVEL]
                            [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                            [--proxy [PROXY]] [--help_summary]
                            [--action ACTION] [--session SESSION]
                            [--deep_clean DEEP_CLEAN] [--key KEY]
                            [--message MESSAGE]

Test the status message passing functions of /status-msg:
- create a session: PUT /status-msg/<new-session-id>
- post message: POST /status-msg/<new-session-id>
- list sessions: GET /status-msg/
- list messages for session: GET /status-msg/<new-session-id>
- delete message: DELETE /status-msg/<new-session-id>/message-id
- delete session: DELETE /status-msg/<new-session-id>/this
- delete all messages in session: DELETE /status-msg/<new-session-id>/all

Example:
./test_status_msg.py

optional arguments:
  -h, --help            show this help message and exit
  --action ACTION
                        Actions can be:
                            run_test    : run a messaging test
                            new         : create new session
                            update      : add message to session, requires --session, --key, --message
                            read        : read message(s) from session, requires --session
                            list        : list messages from session
                            delete      : delete message, all messages using session/all or session using session/this
  --session SESSION     explicit session or session/message-id
  --deep_clean DEEP_CLEAN
                        remove all messages and all sessions
  --key KEY             how to key the message
  --message MESSAGE     message to include

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_bare_argparse found in /lanforge-scripts/py-json/LANfo
  --mgr MGR             hostname for where LANforge GUI is running
  --mgr_port MGR_PORT   port LANforge GUI HTTP service is running on
  --debug, -d           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80 or https://user:pass@proxy.localnet:3128
  --help_summary        Show summary of what this script does
```

py-scripts/test_wanlink.py

```
usage: test_wanlink.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                       [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                       [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                       [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                       [--proxy [PROXY]]
                       [--debugging DEBUGGING [DEBUGGING ...]]
                       [--debug_log DEBUG_LOG] [--no_cleanup] [--help_summary]
                       [--radio RADIO] [--security SECURITY] [--ssid SSID]
                       [--passwd PASSWD] [--name NAME] [--port_A PORT_A]
                       [--port_B PORT_B] [--rate RATE] [--rate_A RATE_A]
                       [--rate_B RATE_B] [--latency LATENCY]
                       [--latency_A LATENCY_A] [--latency_B LATENCY_B]
                       [--jitter JITTER] [--jitter_A JITTER_A]
```

```
                      [--jitter_B JITTER_B] [--jitter_freq JITTER_FREQ]
                      [--jitter_freq_A JITTER_FREQ_A]
                      [--jitter_freq_B JITTER_FREQ_B] [--drop DROP]
                      [--drop_A DROP_A] [--drop_B DROP_B]

optional arguments:
  -h, --help            show this help message and exit
  --name NAME           The name of the wanlink
  --port_A PORT_A       Endpoint A
  --port_B PORT_B       Endpoint B
  --rate RATE           The maximum rate of transfer at both endpoints (bits/s)
  --rate_A RATE_A       The max rate of transfer at endpoint A (bits/s)
  --rate_B RATE_B       The maximum rate of transfer (bits/s)
  --latency LATENCY     The delay of both ports
  --latency_A LATENCY_A
                        The delay of port A
  --latency_B LATENCY_B
                        The delay of port B
  --jitter JITTER       The max jitter of both ports (ms)
  --jitter_A JITTER_A   The max jitter of port A (ms)
  --jitter_B JITTER_B   The max jitter of port B (ms)
  --jitter_freq JITTER_FREQ
                        The jitter frequency of both ports (%)
  --jitter_freq_A JITTER_FREQ_A
                        The jitter frequency of port A (%)
  --jitter_freq_B JITTER_FREQ_B
                        The jitter frequency of port B (%)
  --drop DROP           The drop frequency of both ports (%)
  --drop_A DROP_A       The drop frequency of port A (%)
  --drop_B DROP_B       The drop frequency of port B (%)

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
                         - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key
```

py-scripts/throughput_qos.py

```
usage: throughput_QOS.py [-h] [--mgr MGR] [--mgr_port MGR_PORT]
                         [-u UPSTREAM_PORT] [--num_stations NUM_STATIONS]
                         [--test_id TEST_ID] [-d] [--log_level LOG_LEVEL]
                         [--lf_logger_config_json LF_LOGGER_CONFIG_JSON]
                         [--proxy [PROXY]]
                         [--debugging DEBUGGING [DEBUGGING ...]]
                         [--debug_log DEBUG_LOG] [--no_cleanup]
                         [--help_summary] [--radio RADIO]
                         [--security SECURITY] [--ssid SSID] [--passwd PASSWD]
                         [--mode MODE] [--traffic_type TRAFFIC_TYPE]
                         [--download DOWNLOAD] [--upload UPLOAD]
                         [--test_duration TEST_DURATION] [--create_sta]
                         [--sta_names STA_NAMES] [--ap_name AP_NAME]
                         [--bands BANDS] [--tos TOS] [--ssid_2g SSID_2G]
                         [--security_2g SECURITY_2G] [--passwd_2g PASSWD_2G]
                         [--ssid_6g SSID_6G] [--security_6g SECURITY_6G]
                         [--passwd_6g PASSWD_6G] [--ssid_5g SSID_5G]
                         [--security_5g SECURITY_5G] [--passwd_5g PASSWD_5G]
                         [--radio_2g RADIO_2G] [--radio_5g RADIO_5G]
                         [--radio_6g RADIO_6G]


        NAME: throughput_qos.py

        PURPOSE: throughput_qos.py will create stations, layer3 cross connections and allows user to run the qos traffic
        with particular tos on 2.4GHz and 5GHz bands in upload, download directions.

        EXAMPLE-1:
        Command Line Interface to run download scenario with tos : Voice , bands : 2.4GHz
        python3 throughput_qos.py --ap_name Cisco --mgr 192.168.209.223 --mgr_port 8080 --num_stations 32 --radio_2g wiphy0
        --ssid_2g Cisco --passwd_2g cisco@123 --security_2g wpa2 --bands 2.4g --upstream eth1 --test_duration 1m
        --download 1000000 --upload 0 --traffic_type lf_udp --tos "VO" --create_sta

        EXAMPLE-2:
        Command Line Interface to run download scenario with tos : Voice and Video , bands : 5GHz
        python3 throughput_qos.py --ap_name Cisco --mgr 192.168.209.223 --mgr_port 8080 --num_stations 32 --radio_5g wiphy1
        --ssid_5g Cisco --passwd_5g cisco@123 --security_5g wpa2 --bands 5g --upstream eth1 --test_duration 1m
```

```
        --download 1000000 --upload 0 --traffic_type lf_tcp --tos "VO,VI" --create_sta

        EXAMPLE-3:
        Command Line Interface to run download scenario with tos : Voice and Video , bands : 6GHz
        python3 throughput_qos.py --ap_name Cisco --mgr 192.168.209.223 --mgr_port 8080 --num_stations 32 --radio_6g wiphy1
        --ssid_6g Cisco --passwd_6g cisco@123 --security_6g wpa2 --bands 6g --upstream eth1 --test_duration 1m
        --download 1000000 --upload 0 --traffic_type lf_tcp --tos "VO,VI" --create_sta

        EXAMPLE-4:
        Command Line Interface to run upload scenario with tos : Background, Besteffort, Video and Voice , bands : 2.4GHz and 5GHz
        python3 throughput_qos.py --ap_name Cisco --mgr 192.168.209.223 --mgr_port 8080 --num_stations 64 --radio_2g wiphy0
        --ssid_2g Cisco --passwd_2g cisco@123 --security_2g wpa2 --radio_5g wiphy1 --ssid_5g Cisco --passwd_5g cisco@123
        --security_5g wpa2 --bands both --upstream eth1 --test_duration 1m --download 0 --upload 1000000
        --traffic_type lf_udp --tos "BK,BE,VI,VO" --create_sta

        EXAMPLE-5:
        Command Line Interface to run upload scenario with tos : Background, Besteffort, Video and Voice , bands : 2.4GHz and 5GHz
        python3 throughput_qos.py --ap_name Cisco --mgr 192.168.209.223 --mgr_port 8080 --num_stations 64 --radio_2g wiphy0
        --ssid_2g Cisco --passwd_2g [BLANK] --security_2g open --radio_5g wiphy1 --ssid_5g Cisco --passwd_5g [BLANK]
        --security_5g open --bands both --upstream eth1 --test_duration 1m --download 0 --upload 1000000
        --traffic_type lf_udp --tos "BK,BE,VI,VO" --create_sta

        EXAMPLE-6:
        Command Line Interface to run bi-directional scenario with tos : Video and Voice , bands : 6GHz
        python3 throughput_qos.py --ap_name Cisco --mgr 192.168.209.223 --mgr_port 8080 --num_stations 32 --radio_6g wiphy1
        --ssid_6g Cisco --passwd_6g cisco@123 --security_6g wpa2 --bands 6g --upstream eth1 --test_duration 1m
        --download 1000000 --upload 10000000 --traffic_type lf_udp --tos "VO,VI" --create_sta

        SCRIPT_CLASSIFICATION :  Test

        SCRIPT_CATEGORIES:   Performance,  Functional, Report Generation

        NOTES:
        1.Use './throughput_qos.py --help' to see command line usage and options
        2.Please pass tos in CAPITALS as shown :"BK,VI,BE,VO" Eg : --tos "BK,BE,VO,VI"
        3.Please enter the download or upload intended rate in bps
        4.For running the test with --bands both, the number of stations created on each band will be based on entered --num_stati
        Eg: if --num_stations 64 is given then 32 stations will be created on 2.4GHz and 32 stations will be created on 5GHz band.

        STATUS: BETA RELEASE

        VERIFIED_ON:
        Working date - 03/08/2023
        Build version - 5.4.6
        kernel version - 6.2.16+

        License: Free to distribute and modify. LANforge systems must be licensed.
        Copyright 2023 Candela Technologies Inc.

optional arguments:
  -h, --help            show this help message and exit
  --mode MODE           Used to force mode of stations
  --traffic_type TRAFFIC_TYPE
                        Select the Traffic Type [lf_udp, lf_tcp]
  --download DOWNLOAD   --download traffic load per connection (download rate)
  --upload UPLOAD       --upload traffic load per connection (upload rate)
  --test_duration TEST_DURATION
                        --test_duration sets the duration of the test
  --create_sta          Used to force a connection to a particular AP
  --sta_names STA_NAMES
                        Used to force a connection to a particular AP
  --ap_name AP_NAME     AP Model Name
  --bands BANDS         used to run on multiple radio bands,can be used with multiple stations
  --tos TOS             Enter the tos. Example1 : "BK,BE,VI,VO" , Example2 : "BK,VO", Example3 : "VI"
  --ssid_2g SSID_2G     ssid for  2.4Ghz band
  --security_2g SECURITY_2G
                        security type for  2.4Ghz band
  --passwd_2g PASSWD_2G
                        password for 2.4Ghz band
  --ssid_6g SSID_6G     ssid for  6Ghz band
  --security_6g SECURITY_6G
                        security type for  6Ghz band
  --passwd_6g PASSWD_6G
                        password for 6Ghz band
  --ssid_5g SSID_5G     ssid for  5Ghz band
  --security_5g SECURITY_5G
                        security type  for  5Ghz band
  --passwd_5g PASSWD_5G
                        password for  5Ghz band
  --radio_2g RADIO_2G   radio which supports 2.4G bandwidth
  --radio_5g RADIO_5G   radio which supports 5G bandwidth
  --radio_6g RADIO_6G   radio which supports 6G bandwidth

arguments with PRE-DEFINED DEFAULTS, arguments & defaults defined by create_basic_argparse found in /lanforge-scripts/py-json/LANf
  --mgr MGR, --lfmgr MGR
                        hostname for where LANforge GUI is running
  --mgr_port MGR_PORT, --port MGR_PORT
                        port LANforge GUI HTTP service is running on
  -u UPSTREAM_PORT, --upstream_port UPSTREAM_PORT
                        non-station port that generates traffic: <resource>.<port>, e.g: 1.eth1
  --num_stations NUM_STATIONS
                        Number of stations to create
  --test_id TEST_ID     Test ID (intended to use for ws events)
  -d, --debug           Enable debugging
  --log_level LOG_LEVEL
                        Set logging level: debug | info | warning | error | critical
  --lf_logger_config_json LF_LOGGER_CONFIG_JSON
                        --lf_logger_config_json <json file> , json configuration of logger
  --proxy [PROXY]       Connection proxy like http://proxy.localnet:80
                         or https://user:pass@proxy.localnet:3128
  --debugging DEBUGGING [DEBUGGING ...]
                        Indicate what areas you would like express debug output:
                         - digest - print terse indications of lanforge_api calls
                         - json - print url and json data
                         - http - print HTTP headers
                         - gui - ask the GUI for extra debugging in responses
                         - method:method_name - enable by_method() debugging (if present)
```

```
                        - tag:tagname - enable matching by_tag() debug output
  --debug_log DEBUG_LOG
                        Specify a file to send debug output to
  --no_cleanup          Do not cleanup before exit
  --help_summary        Show summary of what this script does

arguments with NO PRE-DEFINED DEFAULTS, arguments defined by create_basic_argparse found in /lanforge-scripts/py-json/LANforge/lfc
  --radio RADIO         radio EID, e.g: 1.wiphy2
  --security SECURITY   WiFi Security protocol: < open | wep | wpa | wpa2 | wpa3 >
  --ssid SSID           WiFi SSID for script objects to associate to
  --passwd PASSWD, --password PASSWD, --key PASSWD
                        WiFi passphrase/password/key

           Create stations and endpoints and runs L3 traffic with various IP type of service(BK |  BE | Video | Voice)
```

py-scripts/tip_station_powersave.py

```
usage: tip_station_powersave.py [-h] [--help_summary]

This script uses filters from realm's PacketFilter class to filter pcap output for specific packets.
Currently it uses a filter for association packets using wlan.fc.type_subtype<=3. It is also using a filter
for QOS Null packets using wlan.fc.type_subtype==44. Both filters are also looking for the existence of
either the station MAC or the AP MAC in wlan.addr
These are returned as an array of lines from the output in the format
$subtype $mac_addresses $wlan.fc.pwrmgt

#Currently, this test can only be applied to UDP connections


optional arguments:
  -h, --help      show this help message and exit
  --help_summary  Show summary of what this script does

       tip_station_powersave.py
```

py-scripts/update_dependencies.py

```
usage: update_dependencies.py [-h] [--create_venv] [--venv_path VENV_PATH]
                              [--destroy_venv [DESTROY_VENV]] [--only_remove]
                              [--use_python USE_PYTHON] [--symlink]
                              [--no_symlink] [--do_pip_upgrade]
                              [--help_summary]

NAME: update_dependencies.py
PURPOSE:  Installs python3 script package dependencies
OUTPUT: List of successful and unsuccessful installs
NOTES: Run this as lanforge user (not root)

The lanforge-scripts/py-scripts and lanforge-scripts/py-json collection require
a number of Pypi libraries. This script installs those libraries or creates
a virtual environment for those libraries. This script has been updated to
detect PEP 668 externally-managed libraries; in the presence of the externally-managed
condition, a virtual environment will be created in $home/scripts/venv-$3.8.
This script will update a symlink $home/lanforge/venv to default virtual environment
as necessary.

optional arguments:
  -h, --help            show this help message and exit
  --create_venv, -c     Create a virtual environment named $home/scritps/venv-3.8 by default. Will create a symlink $home/scripts/
  --venv_path VENV_PATH, --venv VENV_PATH
                        specify the path of the virtual environment to create. Default location is $home/scripts/venv-3.8, and sym
  --destroy_venv [DESTROY_VENV], --remove_venv [DESTROY_VENV]
                        Remove the named python virtual environment. May be used in conjunction with --create_venv to remove the n
  --only_remove, --remove_only
                        Stop after removing virtual environment. Use with --destroy_venv. Will not upgrade pip. Will remove defaul
  --use_python USE_PYTHON, --python USE_PYTHON
                        Specify the full path of the desired version of python to create the virtual environment with. If not spec
  --symlink, --link     Creates a symlink to the created virtual environment. Use with --create_venv. If no venv path or python ve
  --no_symlink, --nosymlink, --nolink
                        Do not create the $home/scripts/venv symlink.
  --do_pip_upgrade, --upgrade_pip
                        The command `sudo pip3 install --upgrade pip` will be run before the virtual environment is created. Requi
  --help_summary        Show summary of what this script does

Examples:
* Install on Fedora:
    ./update_dependencies.py
* Install on Python 3.11+ Externally Managed system:
    ./update_dependencies.py --create_venv
    (This creates a symlink at /home/lanforge/scripts/venv)
* Install venv with a name:
    ./update_dependencies.py --create_venv --venv_path v311
* Install venv to a specific directory:
    ./update_dependencies.py --create_venv --venv_path /usr/local/venvs/lf548* Remove a venv and stop:
    ./update_dependencies.py --destroy_venv v311 --only_remove
* Re-create the default virtual environment:
    ./update_dependencies.py --destroy_venv --create_venv* Upgrade system pip3 (use when there are permission errors):
    ./update_dependencies.py --upgrade_pip
```

py-scripts/webGUI_update_dependencies.py

```
usage: update_dependencies.py [-h] [--pyjwt] [--help_summary]

        NAME: webGUI_update_dependencies.py

        PURPOSE:  Installs python3 webGUI package dependencies

        OUTPUT: List of successful and unsuccessful installs

        NOTES: Run this as lanforge user (not root)


optional arguments:
  -h, --help      show this help message and exit
```

```
  --pyjwt         Install PyJWT which is necessary for GhostRequest
  --help_summary  Show summary of what this script does
```

py-scripts/wlan_capacity_calculator.py

```
usage: wlan_capacity_calculator.py [-h] [-sta STATION] [-t TRAFFIC] [-p PHY]
                                   [-e ENCRYPTION] [-q QOS] [-m MAC]
                                   [-b BASIC [BASIC ...]] [-pre PREAMBLE]
                                   [-s SLOT] [-co CODEC] [-r RTS] [-c CTS]
                                   [-d DATA] [-ch CHANNEL] [-gu GUARD]
                                   [-high HIGHEST] [-pl PLCP] [-ip IP]
                                   [-mc MC] [-cw CWIN] [-spa SPATIAL]
                                   [-rc RTSCTS] [--help_summary]

       wlan_capacity_calculator.py
       ------------------------------------------------------------------------

    Example of command line to run(11ac Station):
   ./wlan_capacity_calculator.py
        -sta 11ac
        -t Voice
        -d 9
        -spa 3
        -ch 20
        -gu 800
        -high 1
        -e TKIP
        -q Yes
        -ip 3
        -mc 0
        -b 6 12 24 54
        -m 1518
        -co Greenfield
        -cw 15


optional arguments:
  -h, --help            show this help message and exit
  -sta STATION, --station STATION
                        Enter Station Name : [11abg,11n,11ac](by Default 11abg)
  -t TRAFFIC, --traffic TRAFFIC
                        Enter the Traffic Type : [Data,Voice](by Default Data)
  -p PHY, --phy PHY     Enter the PHY Bit Rate of Data Flow : [1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54](by Default 54)
  -e ENCRYPTION, --encryption ENCRYPTION
                        Enter the Encryption  : [None,  WEP ,  TKIP, CCMP](by Default None)
  -q QOS, --qos QOS     Enter the QoS = : [No,  Yes](by Default [No for 11abg] and [Yes for 11n])
  -m MAC, --mac MAC     Enter the 802.11 MAC Frame  : [Any Value](by Default [106 for 11abg] and [1538 for 11n])
  -b BASIC [BASIC ...], --basic BASIC [BASIC ...]
                        Enter the Basic Rate Set : [1,2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54] (by Default [1 2 5.5 11 6 12] for
  -pre PREAMBLE, --preamble PREAMBLE
                        Enter Preamble value : [ Short, Long, N/A](by Default Short)
  -s SLOT, --slot SLOT  Enter the Slot Time  : [Short,  Long, N/A](by Default Short)
  -co CODEC, --codec CODEC
                        Enter the Codec Type (Voice Traffic): {[ G.711 ,  G.723 ,  G.729]by Default G.723 for 11abg, G.711 for 11n
  -r RTS, --rts RTS     Enter the RTS/CTS Handshake : [No,  Yes](by Default No)
  -c CTS, --cts CTS     Enter the CTS-to-self (protection) : [No,  Yes](by Default No)
  -d DATA, --data DATA  Enter the Data/Voice MCS Index : ['0','1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','1
  -ch CHANNEL, --channel CHANNEL
                        Enter the Channel Bandwidth = : ['20','40'] by Default 40 for 11n and ['20','40','80'] by Default 80 for 1
  -gu GUARD, --guard GUARD
                        Enter the Guard Interval = : ['400','800'] (by Default 400)
  -high HIGHEST, --highest HIGHEST
                        Enter the Highest Basic MCS = : ['0','1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16
  -pl PLCP, --plcp PLCP
                        Enter the PLCP Configuration = : ['Mixed','Greenfield'] (by Default Mixed) for 11n
  -ip IP, --ip IP       Enter the IP Packets per A-MSDU = : ['0','1','2','3','4','5','6','7','8','9','10','11','12','13','14','15'
  -mc MC, --mc MC       Enter the MAC Frames per A-MPDU = : ['0','1','2','3','4','5','6','7','8','9','10','11','12','13','14','15'
  -cw CWIN, --cwin CWIN
                        Enter the CWmin (leave alone for default) = : [Any Value] (by Default 15)
  -spa SPATIAL, --spatial SPATIAL
                        Enter the Spatial Streams  = [1,2,3,4] (by Default 4)
  -rc RTSCTS, --rtscts RTSCTS
                        Enter the RTS/CTS Handshake and CTS-to-self   = ['No','Yes'] (by Default No for 11ac)
  --help_summary        Show summary of what this script does

            This python script calculates the theoretical value of three different stations( 11abg/11n/11ac)
```