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

Create Python Scripts To Test Generic Traffic

Goal: Create a script to test Generic traffic using Realm
Using the `realm.py` library we will write a script that will allow us to automate the creation of stations and generic cross connects. We will also be able to start and stop traffic over the cross connects using the script. Station and Cross Connect creation is covered in the Realm Scripting Cookbook. Requires LANforge 5.4.2.

1. ## Creating The Profile

   A. We will use the factory method `self.local_realm.new_generic_cx_profile()` to create our profile object.

   B. After we have done this we can set a few variables for our traffic:

      A. `gen_cx_profile.type` will determine the type of command to execute.
      Example: `self.cx_profile.type = "lfping"`

      B. `gen_cx_profile.dest` is the destination IP address for the command.
      Example: `self.cx_profile.dest = "127.0.0.1"`

      C. `gen_cx_profile.interval` sets the interval at which the command is run in seconds.
      Example: `self.cx_profile.interval = 1`

      D. Example Generic profile init:

```
class GenTest(LFCliBase):
    def __init__(self, host, port, ssid, security, password, sta_list, name_pre
                 number_template="00000", test_duration="5m", type="lfping", de
                 interval=1, radio="wiphy0",
                 _debug_on=False,
                 _exit_on_error=False,
                 _exit_on_fail=False):
        super().__init__(host, port, _debug=_debug_on, _halt_on_error=_exit_on_
        self.host = host
        self.port = port
        self.ssid = ssid
        self.radio = radio
        self.upstream = upstream
        self.sta_list = sta_list
        self.security = security
        self.password = password

        self.number_template = number_template
        self.name_prefix = name_prefix
        self.test_duration = test_duration

        self.local_realm = realm.Realm(lfclient_host=self.host, lfclient_port=s
        self.cx_profile = self.local_realm.new_generic_cx_profile()
        self.cx_profile.type = type
        self.cx_profile.dest = dest
        self.cx_profile.interval = interval

    # Station Profile init
```

2. ## Starting Traffic

   A. To start the traffic we can use the `gen_cx_profile.start_cx()` method. To stop the traffic we can use the `gen_cx_profile.stop_cx()` method.

B. Example start and build method:

```
def build(self):
    self.station_profile.use_security(self.security, self.ssid, self.password)
    self.station_profile.set_number_template(self.number_template)
    print("Creating stations")
    self.station_profile.set_command_flag("add_sta", "create_admin_down", 1)
    self.station_profile.set_command_param("set_port", "report_timer", 1500)
    self.station_profile.set_command_flag("set_port", "rpt_timer", 1)
    self.station_profile.create(radio=self.radio, sta_names_=self.sta_list, debug=self.
    self.cx_profile.create(ports=self.station_profile.station_names, sleep_time=.5)
    self._pass("PASS: Station build finished")
```

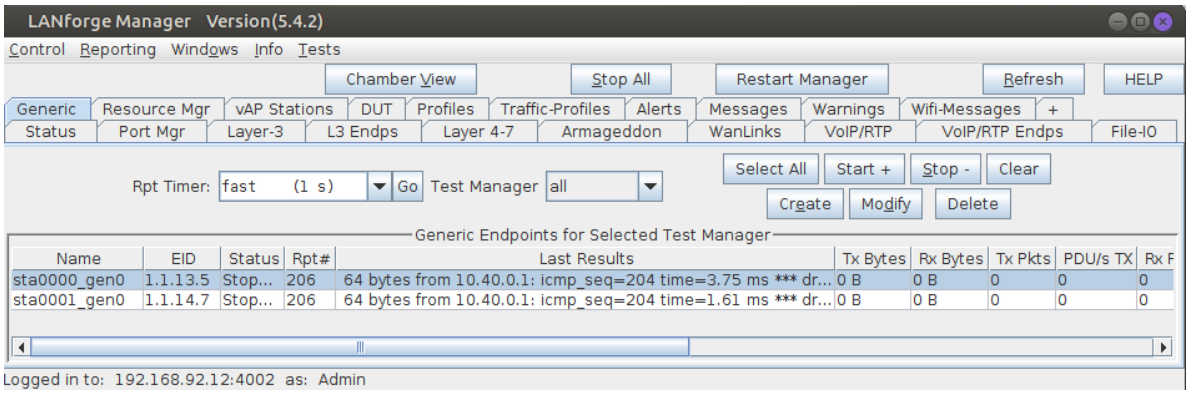
```
def start(self, print_pass=False, print_fail=False):
    self.station_profile.admin_up()
    temp_stas = self.sta_list.copy()
    temp_stas.append(self.upstream)
    if self.local_realm.wait_for_ip(temp_stas):
        self._pass("All stations got IPs", print_pass)
    else:
        self._fail("Stations failed to get IPs", print_fail)
        exit(1)
    cur_time = datetime.datetime.now()
    passes = 0
    expected_passes = 0
    self.cx_profile.start_cx()
    time.sleep(15)
    end_time = self.local_realm.parse_time("30s") + cur_time
    print("Starting Test...")
    while cur_time < end_time:
        cur_time = datetime.datetime.now()
        gen_results = self.json_get("generic/list?fields=name,last+results", debug_=sel
        if gen_results['endpoints'] is not None:
            for name in gen_results['endpoints']:
                for k, v in name.items():
                    if v['name'] in self.cx_profile.created_endp and not v['name'].ends
                        expected_passes += 1
                        if v['last results'] != "" and "Unreachable" not in v['last res
                            passes += 1
                        else:
                            self._fail("%s Failed to ping %s " % (v['name'], self.cx_pr
                            break
        time.sleep(1)
    if passes == expected_passes:
        self._pass("PASS: All tests passed", print_pass)
```

3. ## Examining The Results

A. For `lfping` we can use the last results of the endpoint to determine if the test was successful. An example of this can be seen in our `start` method. The most common errors for `lfping` will either be a blank last result or `Destination Host Unreachable`. Either of these results indicate a failed ping. Successful pings will look like:

```
64 bytes from 10.40.0.1: icmp_seq=1 time=4.55 ms *** drop: 0 (0, 0.000)  rx: 1  fail: 0
```

Results can also be seen in the generic tab in the LANforge Manager:



Double-clicking on an endpoint will allow you to see more specific results as well as the command used by the endpoint. Using the **sync** button will allow you to see updated results.