Creating Endpoint Hunt Scripts with CLI API

**Goal:** Use the CLI to operate the Endpoint Scripting features of the Layer-3 Endpoints you create.

Layer-3 endpoints can manipulate their own transmission features using a variety of internal scripts, known as Endpoint Scripts. Using the `lf_endp_script.pl` CLI script, you can operate those internal endpoints behaviours.

This cookbook talks about Endpoint Scripts and CLI scripts at the same time. In this chapter, if the term script is used, assume Endpoint Script. Additionally, the terms operating and running can also be confusing. To keep the activities distinct, a LANforge user will operate a CLI script from a terminal. The LANforge server will run the Endpoint Script. A CLI script is a user-space perl script that issues CLI commands to a LANforge server. A CLI command is an instruction obeyed by the LANforge server.

**The Forces at Play**

There are a number of subsystems running while we operate an automated Endpoint Script, so let’s review them:

- There will be Layer 3 connect constructed using `lf_firemod.pl`, (Don’t forget: create the endpoints before creating the cross connect.)
- A managed endpoint of that connection will be configured with an Endpoint Script.
- The attending engineer will operate a CLI script that changes state the Layer-3 connection to Running
- The Layer-3 connection starts both endpoints transmitting, one of them starts running its Endpoint Script that sets its transmit parameters.
- Remember, Endpoint Scripts run inside the LANforge server process. CLI scripts run from the client side.

**Let’s Walk Thru Putting One Together**

We recommend starting your first script off by the LANforge GUI to save an endpoint with an Endpoint Script. Next, inspect the LANforge database on the server for the script parameters. Take those parameters and adapt them to the operator’s CLI script.

1. From the Layer-3 tab, open a connection `tutorial-cx`, and navigate to box 2. Click on the Script button,
2. Name your script.

3. Select your Script type, here we choose ScriptHunt.

4. We immediately see the parameters for the script:

5. Let's modify the parameters to match our CLI command example below:
6. In a LANforge terminal, let’s look at /home/Lanforge/DFT/endps.db. We will search for bunny-script and we’ll inspect the resulting CU command.

7. Now we can craft this command into a CLI script. In a CMD window, we can write the formatted CLI script arguments:

```
C:\> .lf_endp_script.pl --mgr jedtest --resource 1 --action set_script --script_type Hunt --script_name bunny-script --endp_name tutorial-CX-A --loops 1 --flags 37120 --private "5000 5000 100000,20000,100000,20,56000,30000,1,100000,60,128,256,512,1024,1280,1460,1472,1514 1472 0,100,300,500,550,600,650,700,750,850,900,950 NONE"
```

In a Linux terminal, we can use double-quotes for quoted script arguments. Using single-quotes will break your command.

```
$ ./lf_endp_script.pl --mgr jedtest --resource 1 --action set_script --script_type Hunt --script_name bunny-script --endp_name tutorial-CX-A --loops 1 --flags 37120 --private '5000 5000 100000,20000,100000,20,56000,30000,1,100000,60,128,256,512,1024,1280,1460,1472,1514 1472 0,100,300,500,550,600,650,700,750,850,900,950 NONE'
```

8. We can start the connection and the Endpoint Script will immediately begin running:

```
lf_endp_script.pl --mgr jedtest --resource 1 --action start_cx --cx_name tutorial-CX
```

9. If the number of loops is set, it will eventually quiesce and stop itself. If we need to stop it and let in flight packets come to rest, we can quiesce it:

```
lf_endp_script.pl --mgr jedtest --resource 1 --action quiesce_cx --cx_name tutorial-CX
```

We could also use action stop_cx to immediately stop the connection.

10. If you have a LANforge GUI running, the EndPoint Script report will automatically display in a GUI window as soon as the connection starts. To display it to the terminal, you need to enable debug output:

```
lf_endp_script.pl --action show_report --endp_name tutorial-CX-A --quiet no
```

Or to save it to a text file:

```
lf_endp_script.pl --action show_report --endp_name tutorial-CX-A --quiet no > /home/lanforge/Documents/report.txt
```

11. To remove the script:

```
lf_endp_script.pl --action remove_script --endp_name tutorial-CX-A
```

**At the CLI Command Level**

**Review of the set_script CLI command**

We have covered creating endpoints in earlier cookbooks. The perl script `lf_endp_script.pl` was created to modify endpoints and operate their Endpoint Scripts. That script is using the `set_script` CLI command (documented here). A call to it looks like:

```
set_script tutorial-cx-A bunny-script 37120 ScriptHunt '...' ALL 20
```

**Endpoint Scripting Uses Large Parameters**

That vague ‘...’ section is the `private` parameter which is a parameter list each script type requires. The `private` parameter combines a series of constraints (sub-parameters). For the `ScriptHunt`, we might use:

```
run_duration pause_duration constraints payload_sizes_a payload_sizes_b attenuations attenuator
```
| drops, jitter_us, latency_us, max_steps, start_rate, accuracy, is_bps, max_tx_slowdown |
| 5000, 5000 |
| 100000, 20000, 100000, 20, 56000, 30000, 1, 100000 |
| 60, 128, 256, 512, 1024, 1280, 1460, 1472, 1514 |
| 0, 100, 300, 500, 550, 600, 650, 700, 750, 850, 900, 950 |
| 1472 |
| 1.1.14 |

Accuracy is also Threshold. max_tx_slowdown is also Undermin. The result is a very long line that has to be surrounded the the CU level by one pair of single quotes:

`5000 5000 100000, 20000, 100000, 20, 56000, 30000, 1, 100000, 60, 128, 256, 512, 1024, 1280, 1460, 1472, 1514 0, 100, 300, 500, 550, 600, 650, 700, 750, 850, 900, 950 NONE`