# TRex Documentation

https://trex-tgn.cisco.com/trex/doc/

Stateless mode

<https://trex-tgn.cisco.com/trex/doc/trex_manual.html>

Config file: /etc/trex\_cfg.yaml

cp cfg/simple\_cfg.yaml /etc/trex\_cfg.yaml

./dpdk\_setup\_ports.py -s

./dpdk\_setup\_ports.py -i

./dpdk\_setup\_ports.py -t

./dpdk\_setup\_ports.py -c <TRex interface 0> <TRex interface 1> ..

./t-rex-64 -f cap2/dns.yaml **--lm 1 --lo** -l 1000 -d 100

Dummy ports:

interfaces: ['07:00.0', 'dummy', 'dummy', '8a:00.0']

./t-rex-64 -f cap2/dns.yaml -c 4 -m 1 -d 10

./bp-sim-64-debug -f cap2/dns.yaml -o my.erf -v 3

wireshark my.erf

trex>service

trex>l2 --help

usage: port [-h] --port PORT --dst DST\_MAC

Configures a port **in** L2 mode

optional arguments:

-h, --help show this **help** message and **exit**

--port PORT, -p PORT **source** port **for** the action

--dst DST\_MAC Configure destination MAC address

trex(service)>l2 -p 0 --dst 6A:A7:B5:3A:4E:FF

Setting port 0 **in** L2 mode: [SUCCESS]

trex>service --off

trex>service

trex>l2 --help

usage: port [-h] --port PORT --dst DST\_MAC

Configures a port **in** L2 mode

optional arguments:

-h, --help show this **help** message and **exit**

--port PORT, -p PORT **source** port **for** the action

--dst DST\_MAC Configure destination MAC address

trex(service)>l2 -p 0 --dst 6A:A7:B5:3A:4E:FF

Setting port 0 **in** L2 mode: [SUCCESS]

trex>service --off

**Example Of switching between Service and Normal modes: API**

client.**set\_service\_mode**(ports = [0, 1], enabled = True)

client.**set\_service\_mode**(ports = [0, 1], enabled = False)

Connect from a remote server:

Statrt Server:

[bash]>sudo ./t-rex-64 -i

[bash]>trex-console -s csi-kiwi-02

-s option is used to specify trex server

**TRex command line examples**

**Simple HTTP 1Gb/sec for 100 sec**

[bash]>sudo ./t-rex-64 -f cap2/simple\_http.yaml -c 4 -m 100 -d 100

**Simple HTTP 1Gb/sec with latency for 100 sec**

[bash]>sudo ./t-rex-64 -f cap2/simple\_http.yaml -c 4 -m 100 -d 100 -l 1000

**SFR 35Gb/sec traffic**

[bash]>sudo ./t-rex-64 -f avl/sfr\_delay\_10\_1g.yaml -c 4 -m 35 -d 100 -p

**SFR 20Gb/sec traffic with latency**

[bash]>sudo ./t-rex-64 -f avl/sfr\_delay\_10\_1g.yaml -c 4 -m 20 -d 100 -l 1000

**SFR ipv6 20Gb/sec traffic with latency**

[bash]>sudo ./t-rex-64 -f avl/sfr\_delay\_10\_1g\_no\_bundeling.yaml -c 4 -m 20 -d 100 -l 1000 --ipv6

**Simple HTTP 1Gb/sec with NAT translation support**

[bash]>sudo ./t-rex-64 -f cap2/simple\_http.yaml -c 4 -m 100 -d 100 -l 1000 --learn-mode 1

**IMIX 1G/sec ,1600 flows**

[bash]>sudo ./t-rex-64 -f cap2/imix\_fast\_1g.yaml -c 4 -m 1 -d 100 -l 1000

**IMIX 1Gb/sec,100K flows**

[bash]>sudo ./t-rex-64 -f cap2/imix\_fast\_1g\_100k.yaml -c 4 -m 1 -d 100 -l 1000

**64bytes ~1Gb/sec,1600 flows**

[bash]>sudo ./t-rex-64 -f cap2/imix\_64.yaml -c 4 -m 1 -d 100 -l 1000

**IMIX support:**

[bash]>sudo ./t-rex-64 -f cap2/imix\_64.yaml -d 1000 -m 40000 -c 4 –p

./stl-sim -f stl/udp\_1pkt\_simple.py -o b.pcap -l 10

./stl-sim -f stl/udp\_1pkt\_simple.py --json

./stl-sim -f stl/udp\_1pkt\_simple.py --yaml

./stl-sim -f stl/udp\_1pkt\_simple.py --pkt

./stl-sim -f my\_yaml.yaml –native

./stl-sim -f /tmp/T9600\_MainLine\_VLAN\_2021\_06\_21\_093437/trex/udp\_dot1q\_1pkt\_simple.py -o /tmp/1udp\_dot1.pcap -l 10

<https://trex-tgn.cisco.com/trex/doc/trex_appendix_fixing_pcaps.html#_centering_the_capture_point>

**Tutorial: Python automation**

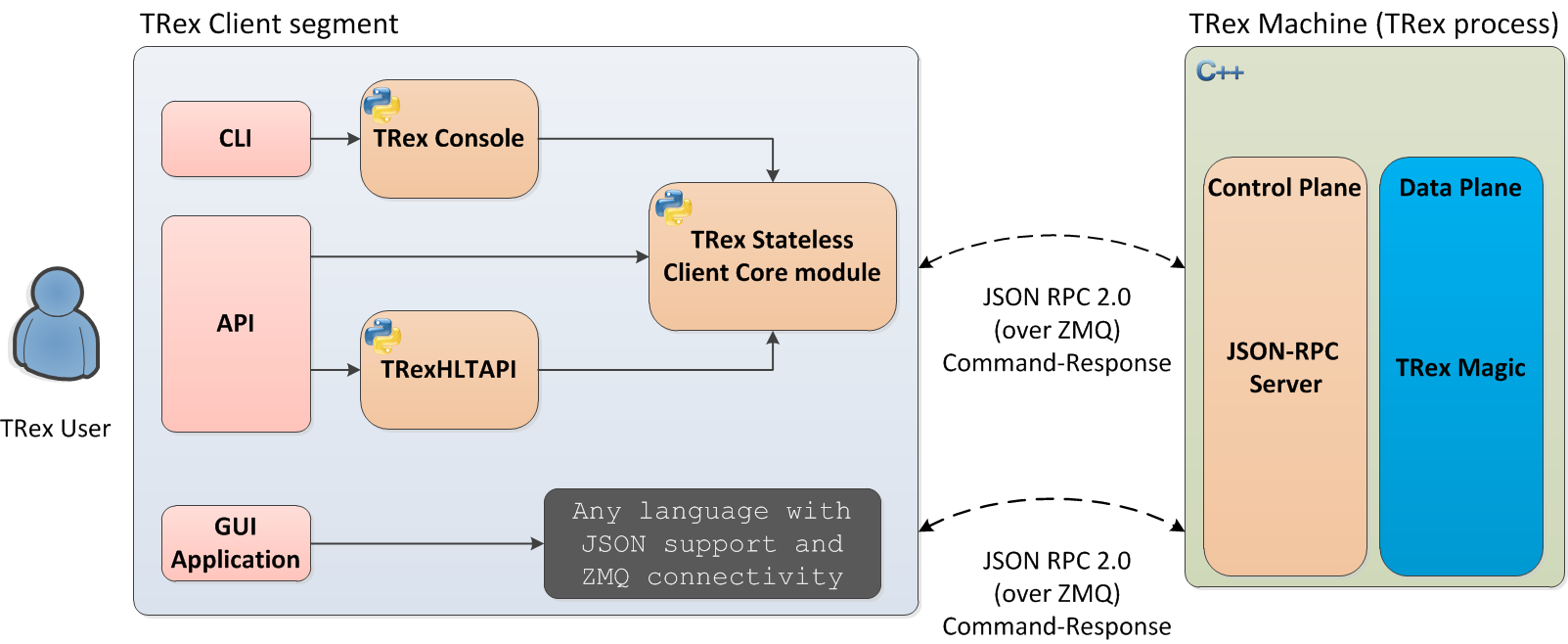
**Goal**

Simple automation test using Python from a local or remote machine.

**Directories**

Python API examples: automation/trex\_control\_plane/interactive/trex/examples/stl

Python API library: automation/trex\_control\_plane/interactive/trex/stl



Stateless API Reference

<https://trex-tgn.cisco.com/trex/doc/cp_stl_docs/api/client_code.html>

<https://trex-tgn.cisco.com/trex/doc/cp_stl_docs/index.html>

example: /opt/trex/v2.89/automation/trex\_control\_plane/interactive/trex/examples/stl/stl\_functional.py

export PYTHONPATH=$PYTHONPATH:/opt/trex/v2.89/automation/trex\_control\_plane/interactive

export STL\_PROFILES\_PATH=/opt/trex/v2.89/stl

export EXT\_LIBS\_PATH=/opt/trex/v2.89/external\_libs

STL\_PROFILES\_PATH: /opt/trex/v2.89/stl

EXT\_LIBS\_PATH: /opt/trex/v2.89/external\_libs

TREX\_EXT\_LIBS /opt/trex/v2.89/trex\_client/external\_libs

#### HLT Python API

Example: <https://github.com/cisco-system-traffic-generator/trex-core/tree/master/scripts/automation/trex_control_plane/interactive/trex/examples/stl/hlt_udp_simple.py>

APIs: <https://trex-tgn.cisco.com/trex/doc/trex_stateless.html#_hlt_supported_arguments_a_id_altapi_support_a>

**Example of configuring L2 mode- Python API**

client.**set\_service\_mode**(port = 0, enabled = True)

client.**set\_l2\_mode**(port = 0, dst\_mac = "6A:A7:B5:3A:4E:FF")

client.**set\_service\_mode**(port = 0, enabled = False)

**Example of configuring L2 mode- Python API**

client.**set\_service\_mode**(port = 0, enabled = True)

client.**set\_l2\_mode**(port = 0, dst\_mac = "6A:A7:B5:3A:4E:FF")

client.**set\_service\_mode**(port = 0, enabled = False)

*# move port 1 to service mode as we want to capture traffic on it*

client.**set\_service\_mode**(ports = 1)

*# start a capture on port 1 Rx side with a limit, a mode and a \*BPF\* filter for any UDP with dst port 53*

capture = client.**start\_capture**(rx\_ports = 1, limit = 100, mode = 'fixed', bpf\_filter = 'udp and dst 53')

*# execute your code here*

*# save the packets to a file or to a list (see the Python API docs)*

client.**stop\_capture**(capture['id'], '/home/mydir/port\_0\_rx.pcap')

*# exit service mode on port 1*

client.**set\_service\_mode**(ports = 1, enabled = False)

## TRex Control Plane Design - Phase 1

<https://trex-tgn.cisco.com/trex/doc/trex_control_plane_design_phase1.html>

<https://trex-tgn.cisco.com/trex/doc/cp_docs/>

# Client Package

Starting from version v1.99 TRex has separated client package included in main directory.

Put it at any place you like, preferably same place as your scripts.

(If it’s not at same place as your scripts, you will need to ensure trex\_client directory is in sys.path)

Un-pack it using command:

tar -xzf trex\_client\_<TRex version>.tar.gz

The client assumes stateful daemon is running:

sudo ./trex\_daemon\_server start

After un-tarring the client package, you can verify basic tests in examples directory out of the box:

cd trex\_client/stf/examples

python stf\_example.py -s <server address>