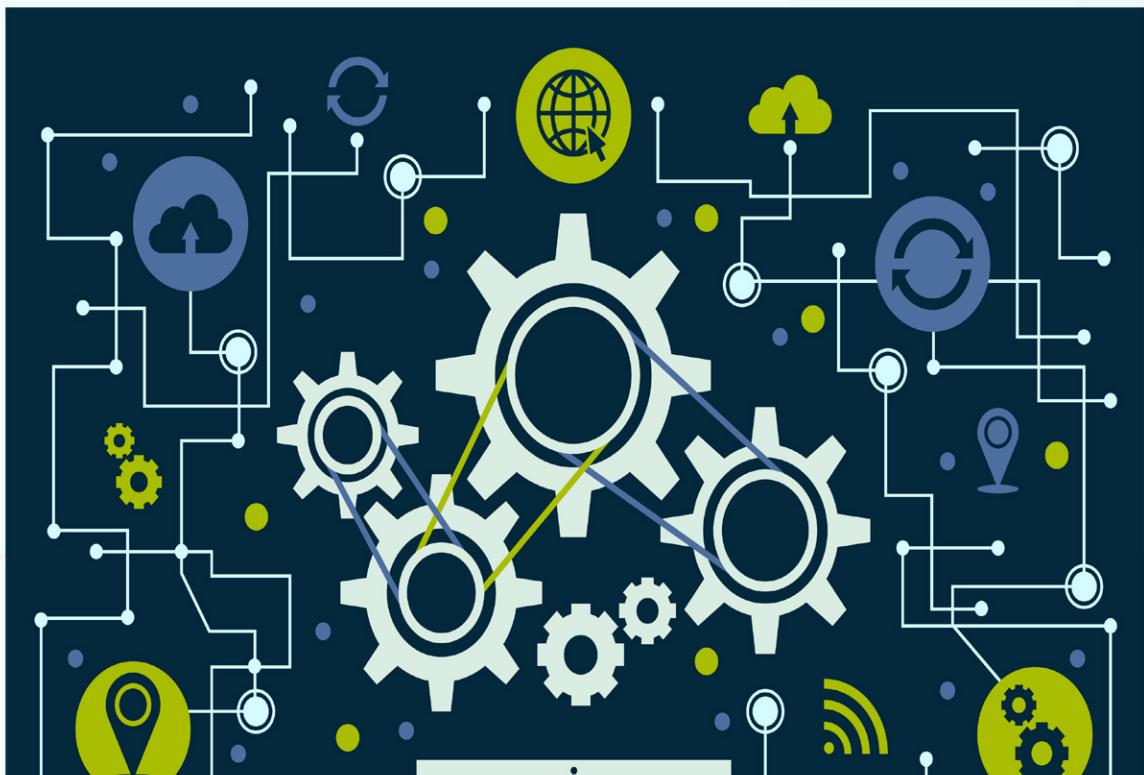


IXNETWORK-CLASSIC QUICK REFERENCE GUIDE



PREFACE

1. Overview.....	2
2. Configure OSPFv2 through GUI.....	2
2.1 Add Chassis and Reserve Ports	3
2.2 Configure Interfaces	4
2.3 Emulate OSPFv2 Protocol	5
2.4 Configure OSPFv2	5
2.5 Create OSPFv2 Route Ranges	6
2.6 Start OSPFv2 Protocol.....	6
2.7 Check Learned LSAs	7
2.8 Configure Traffic	8
2.9 Add Endpoints To Traffic	8
2.10 Edit Packet	9
2.11 Setup Flow Group	9
2.12 Setup Frame Size	10
2.13 Setup Line Rate	10
2.14 Setup Flow Tracking.....	11
2.15 Preview Flow Groups.....	12
2.16 Validate Traffic Items.....	12
2.17 Apply Traffic, Start Traffic and Statistics View.....	13
3. Configure OSPFv2 through Automation	14
3.1 Initialize Environment.....	14
3.2 Add Chassis and Reserve Ports	14
3.3 Configure Ports	15
3.4 Create OSPFv2	16
3.5 Create OSPFv2 Route Ranges	17
3.6 Start Protocols	18
3.7 Check Learned LSAs	19
3.8 Configure Traffic	21
3.9 Start and Stop Traffic	22
3.10 Get Statistics	23
4. Other Utilities	24
4.1 IxNetwork API Documentation Browser.....	24
4.2 Script Gen.....	25
4.3 F1 Option	26
5. To Know More on IxNetwork Classic	27
6. Support	27

1. Overview

- Ix-Network Classic offers the performance and functionality testing of Routers/Switches.
- Provides a powerful, yet easy-to-use, graphical user interface (GUI) that you can use to configure and run complex tests.
- Offers the flexibility to customize the application to meet a wide range of requirements for testing complex network topologies, consisting of thousands of routing or switching devices.
- Emulate millions of routes and reachable hosts within the topology. Provides with the ability to customize millions of traffic flows to stress the data plane performance.
- Create sophisticated configurations using powerful wizards and grid controls in GUI.
- Capable of reporting comprehensive protocol status and detailed per-flow traffic performance metrics.

2. Configure OSPFv2 through GUI

This section visualizes the scenario to configure OSPFv2 protocol through GUI and verify IPv4 traffic. Section includes the following tasks.

- Add chassis and reserve ports.
- Configure OSPFv2 protocol on interfaces. Disable ‘Discard Learned LSAs’ on port 1/1/7 for the peer (1/1/8) to learn LSAs. Create route ranges to advertise prefixes to the peer.
- Start OSPFv2 protocol. Once OSPFv2 protocol sessions UP, check for learned LSAs in the peer (1/1/8) side. Please refer Fig 7.1.
- Create IPv4 unicast traffic (1/1/7 -> 1/1/8). Start traffic and wait for some time for the traffic to flow. Stop the traffic. Verify Tx == Rx packets count from Traffic Item Statistics section.

2.1 Add Chassis and Reserve Ports

Add Chassis and reserve ports.

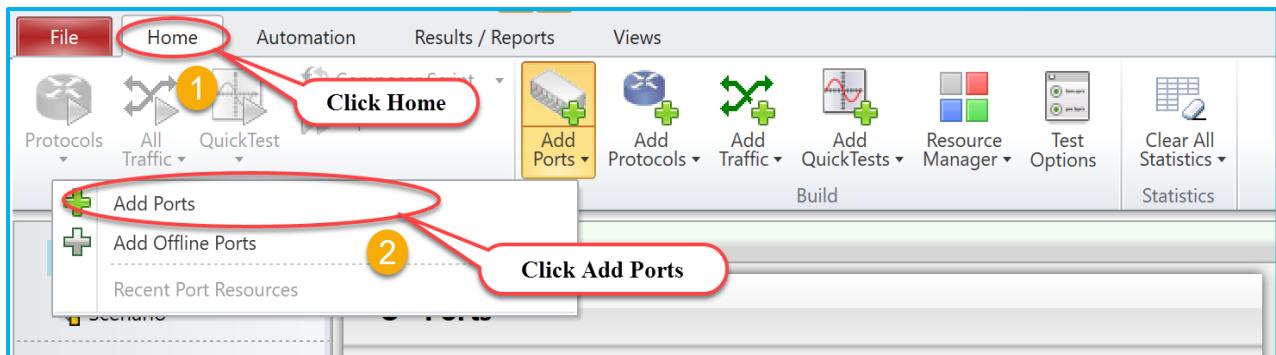


Fig 1.1 Add Ports

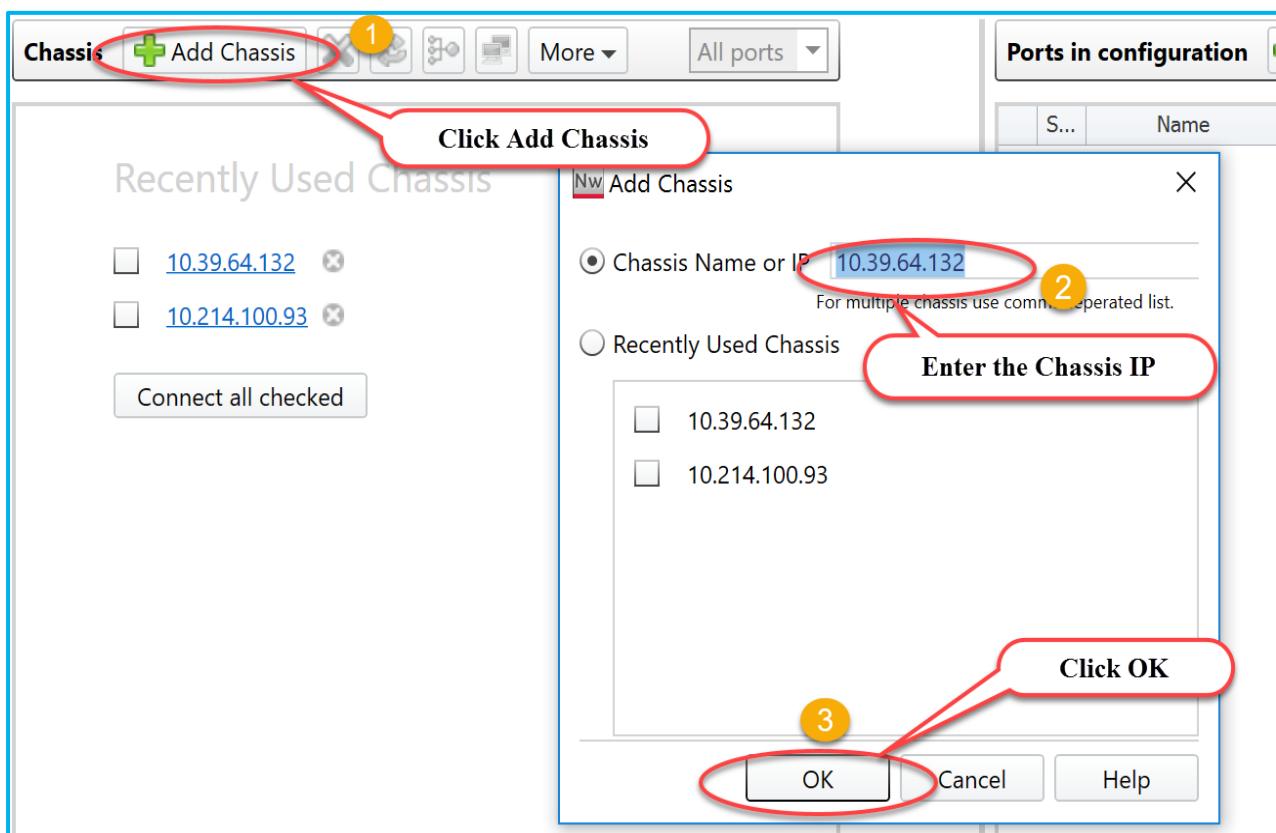


Fig 1.2: Add Chassis to reserve ports

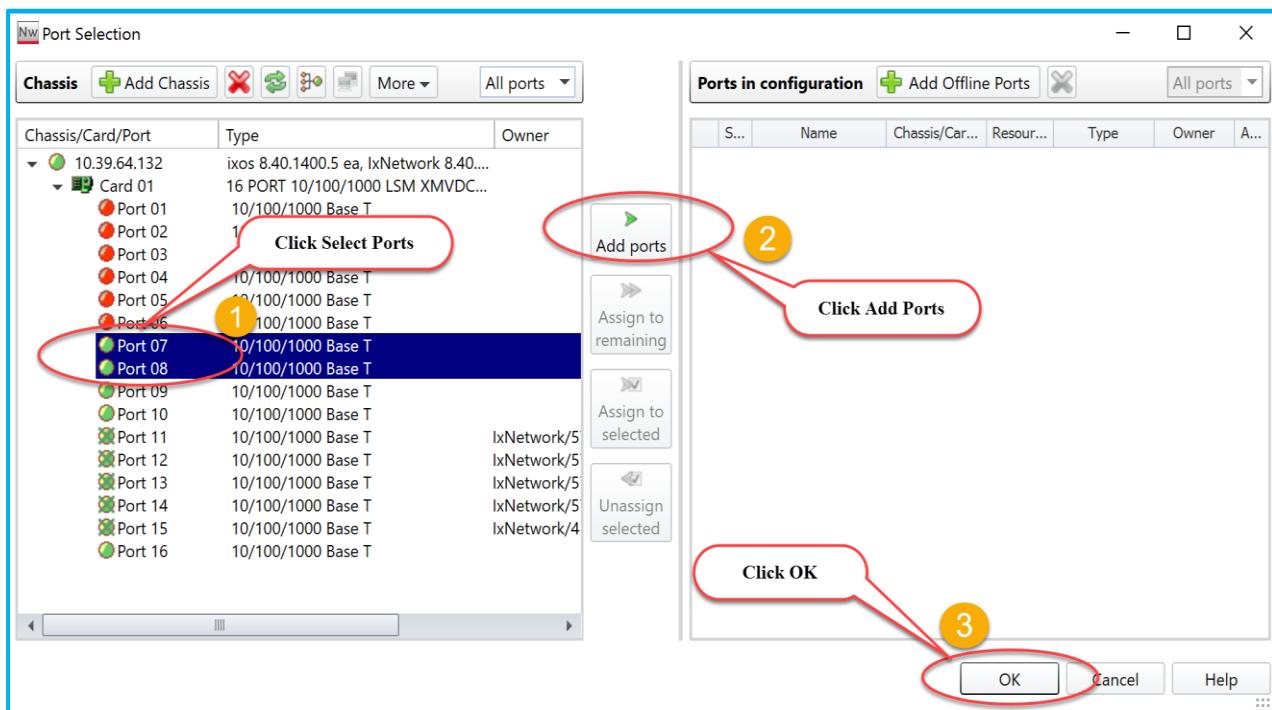


Fig 1.3: Select Ports and Reserve Ports

2.2 Configure Interfaces

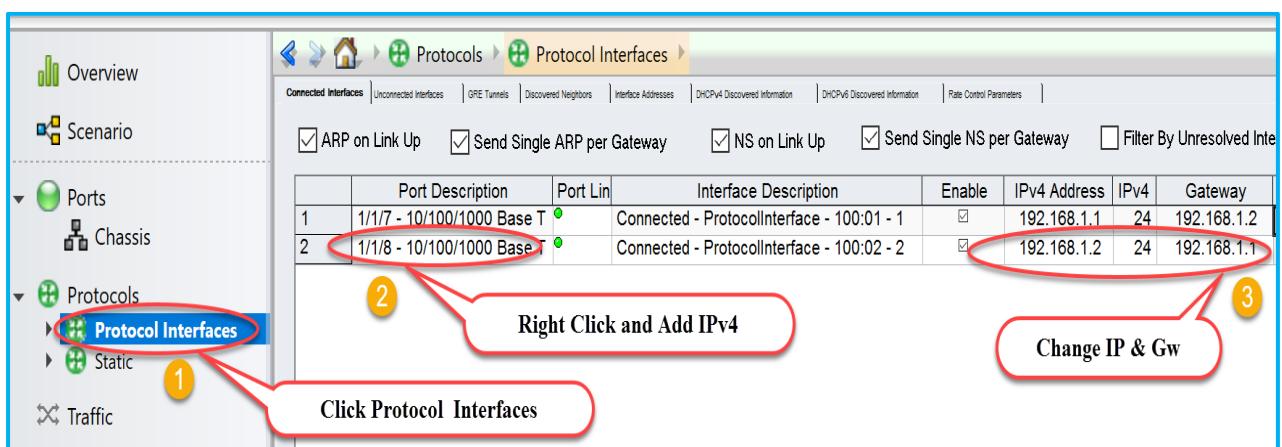


Fig 2.1: Configure Port's attributes

2.3 Emulate OSPFv2 Protocol

Enable OSPFv2 on interfaces.

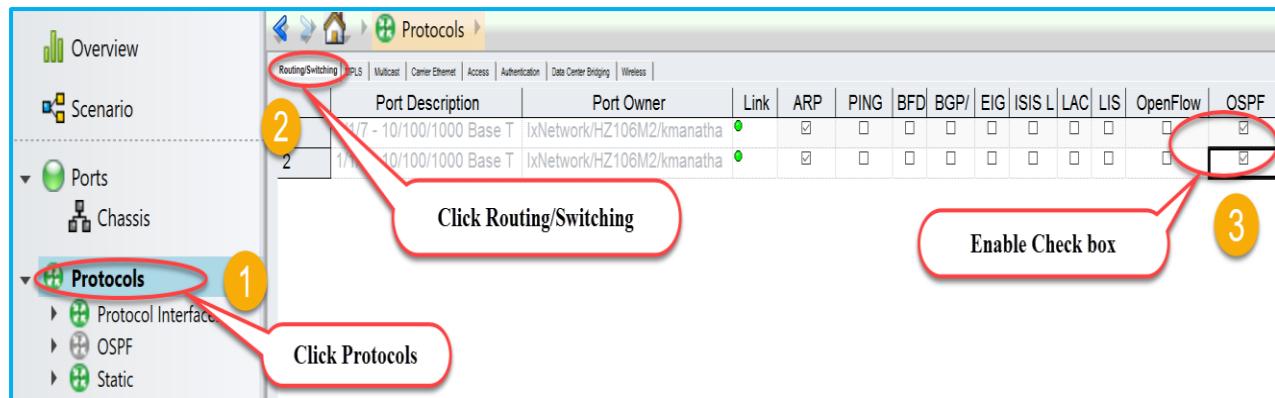


Fig 3.1: Emulate OSPFv2

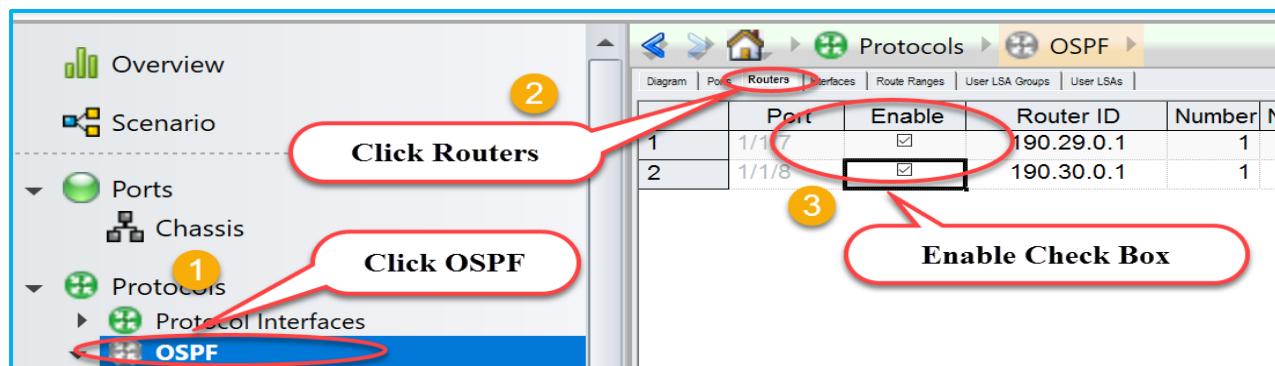


Fig 3.2: Enable OSPFv2 on interfaces

2.4 Configure OSPFv2

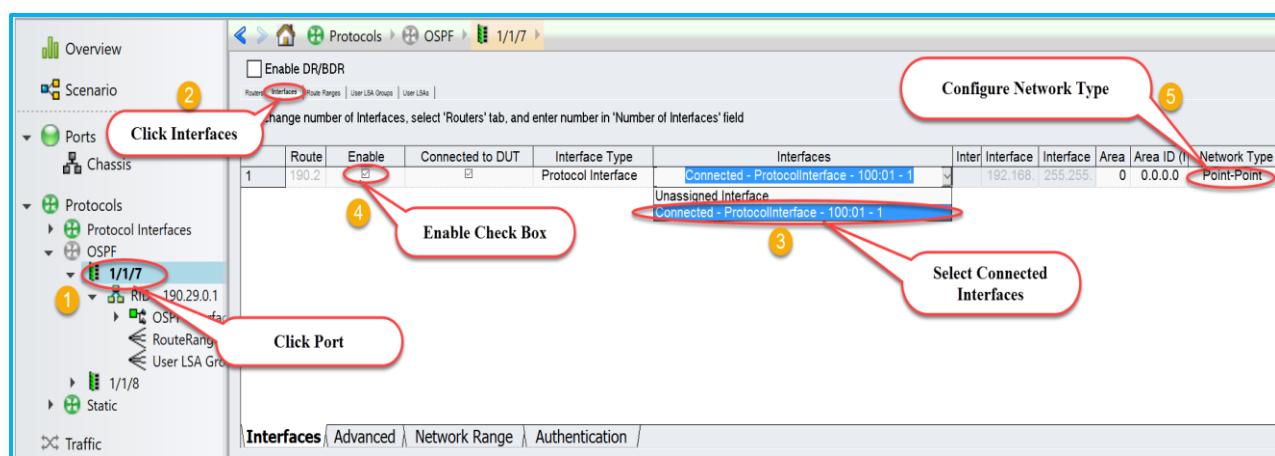


Fig 4.1: Map connected interfaces to configure OSPFv2 attributes

2.5 Create OSPFv2 Route Ranges

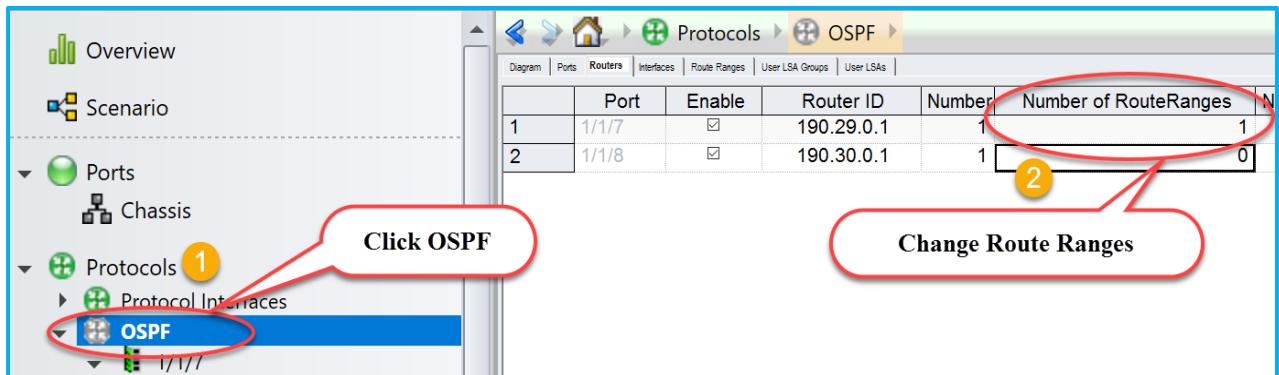


Fig 5.1: Configure number of route ranges required

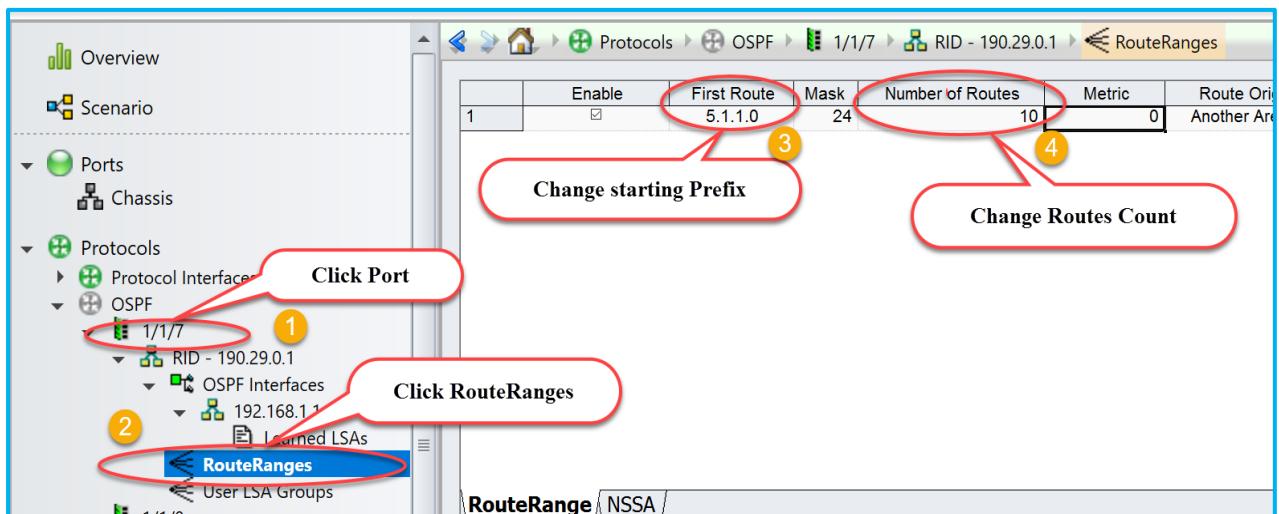


Fig 5.2: Create Ipv4 prefix pool on selected port 1/1/7

2.6 Start OSPFv2 Protocol

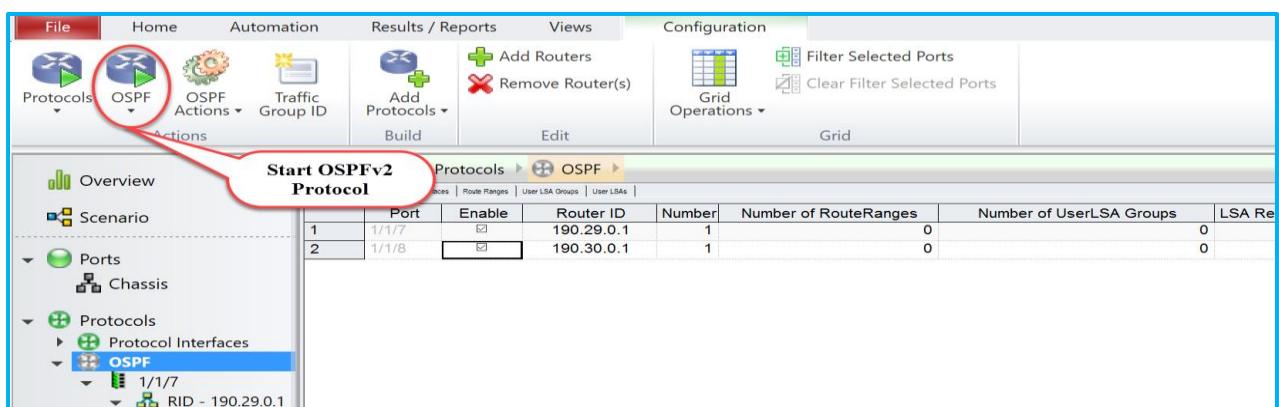


Fig 6.1: Start OSPFv2 protocols

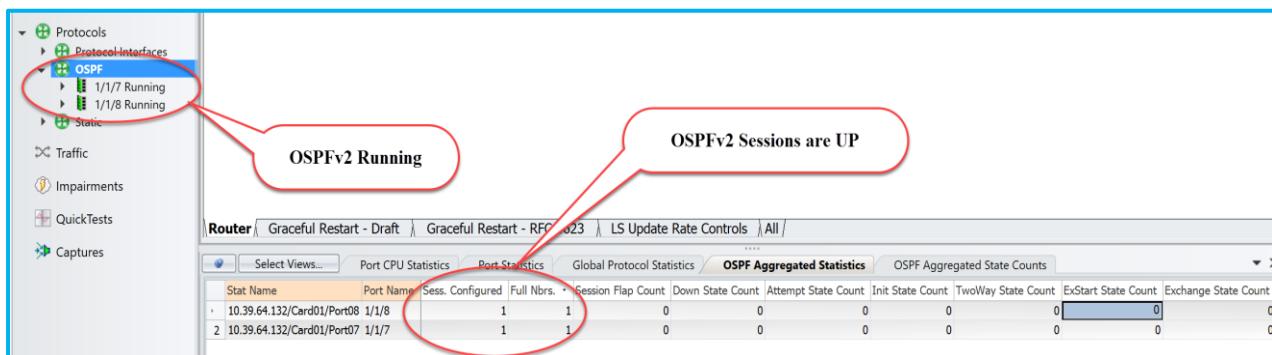


Fig 6.2 OSPFv2 is in Running state and sessions are UP

2.7 Check Learned LSAs

Disable “Discard Learned LSAs” to view learned LSAs on 1/1/8. Please refer Fig 7.1

The top screenshot shows the 'Protocols > OSPF' configuration table. A red circle highlights the 'Discard Learned LSAs' checkbox for port 1, which is checked. A yellow circle labeled '1' highlights the 'OSPF' link in the tree view. A red circle labeled '2' highlights the 'Disabled Discard Learned LSAs' annotation. The bottom screenshot shows the 'Number of LSAs 6' table. A red box highlights the first six rows (LSA IDs 1-6). A yellow circle labeled '2' highlights the 'Learned LSAs From Peer' annotation. The bottom-left tree view shows 'Learned LSAs' for both '1/1/7 Running' and '1/1/8 Running' ports. A red circle labeled '1' highlights the 'Learned LSAs' link for the '1/1/8 Running' port.

	Link State ID	Advertising Router	LSA Type
1	190.29.0.1	190.29.0.1	Router
2	5.1.1.0	190.29.0.1	Summary IP
3	5.1.2.0	190.29.0.1	Summary IP
4	5.1.3.0	190.29.0.1	Summary IP
5	5.1.4.0	190.29.0.1	Summary IP
6	5.1.5.0	190.29.0.1	Summary IP

Fig 7.1 Disable discard Learned LSA and check Learned LSAs on 1/1/8 from peer router 1/1/7

2.8 Configure Traffic

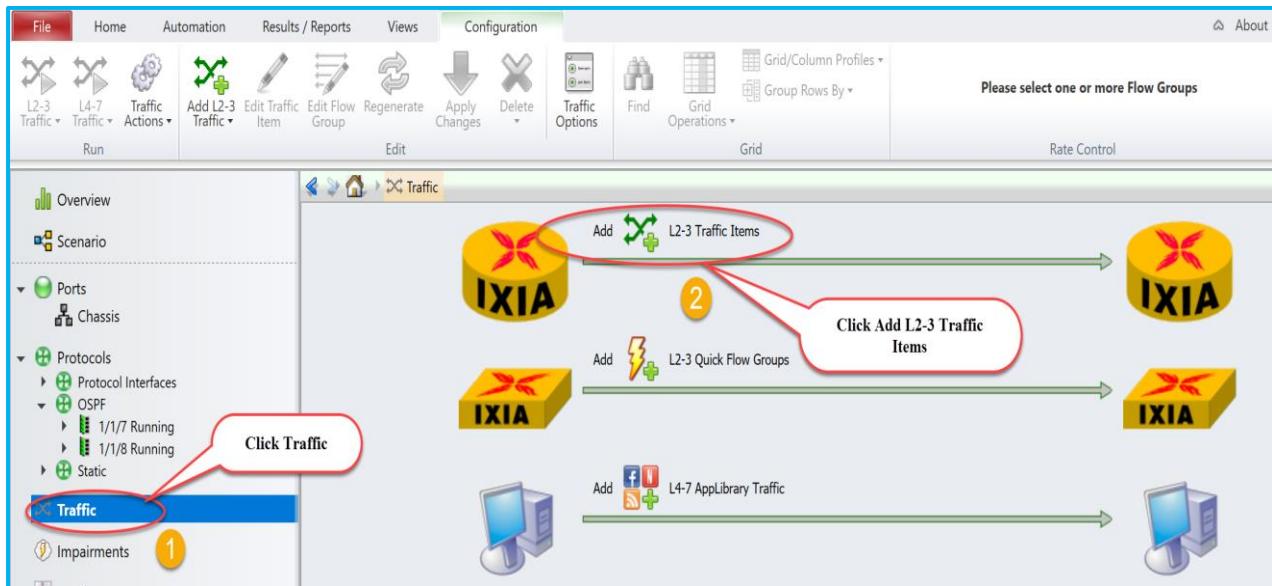


Fig 8.1: Create traffic stream

2.9 Add Endpoints To Traffic

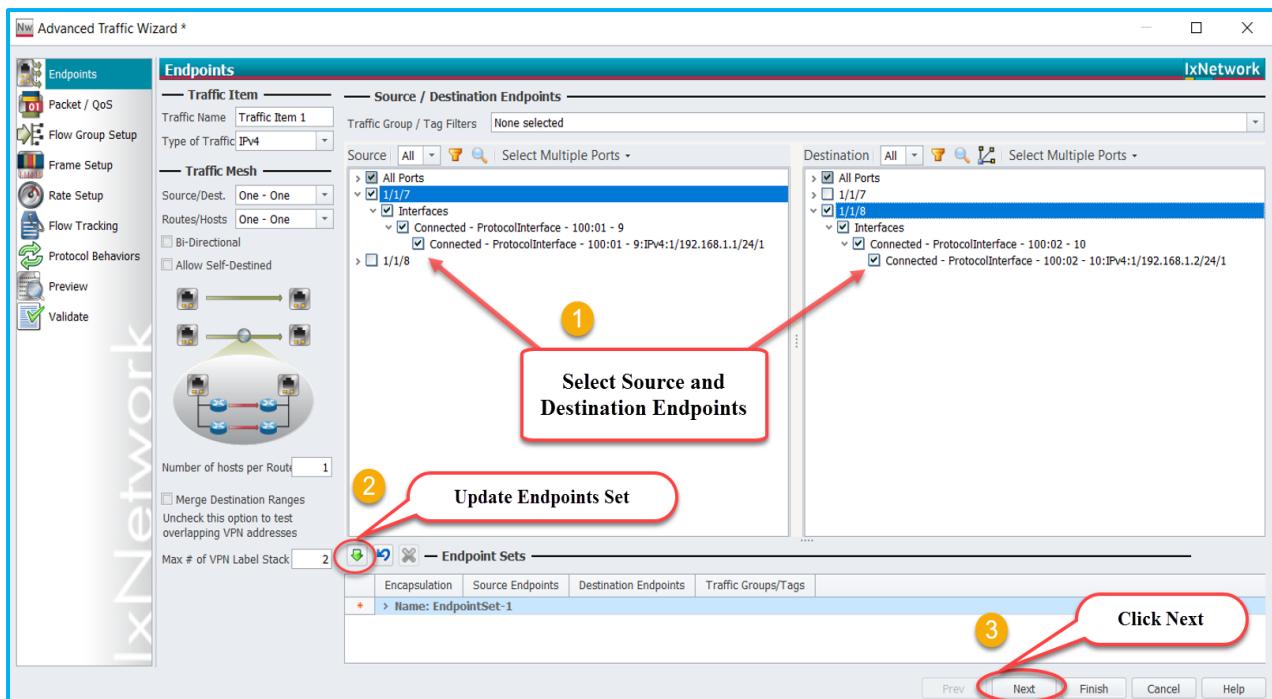


Fig 8.2: Setup source and destination endpoints

2.10 Edit Packet

*Edit Packet and Flow Group Setup are optional.

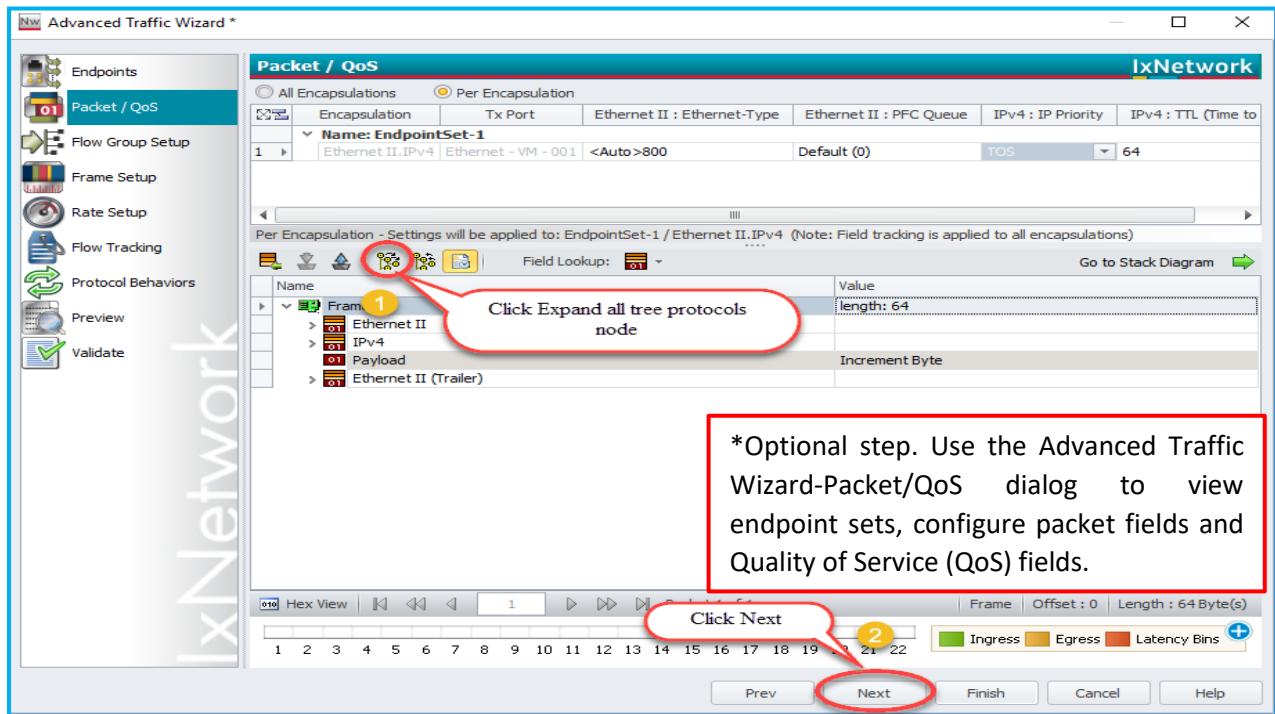


Fig 8.3: Edit packet header

2.11 Setup Flow Group

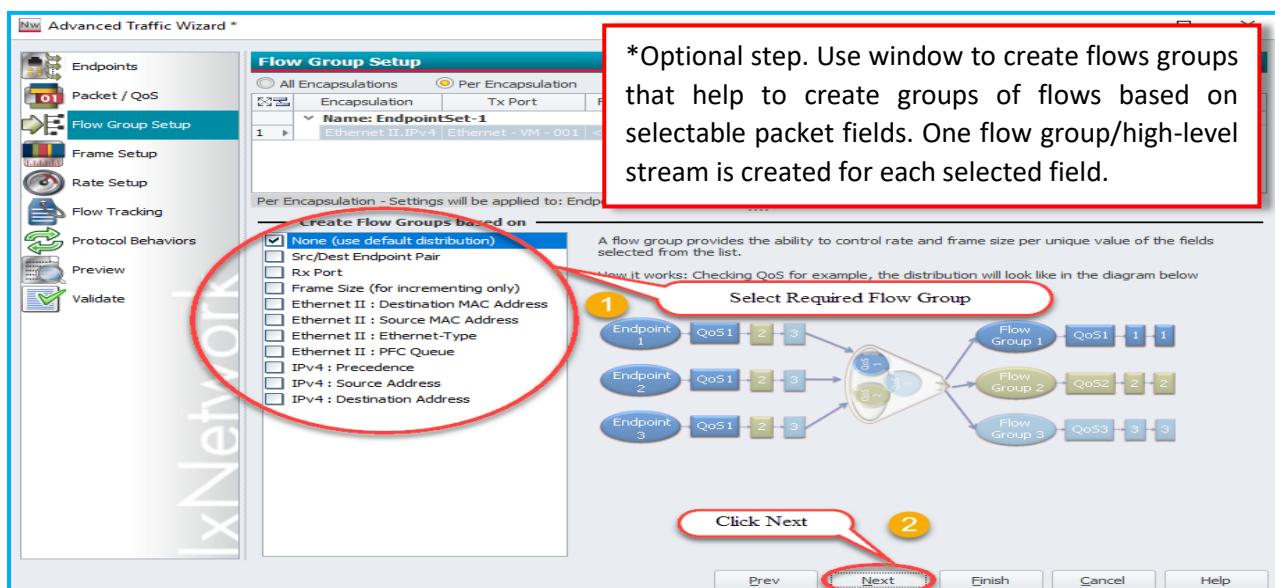


Fig 8.4: Setup flow group

2.12 Setup Frame Size

*Setup frame Size and Line rate are optional.

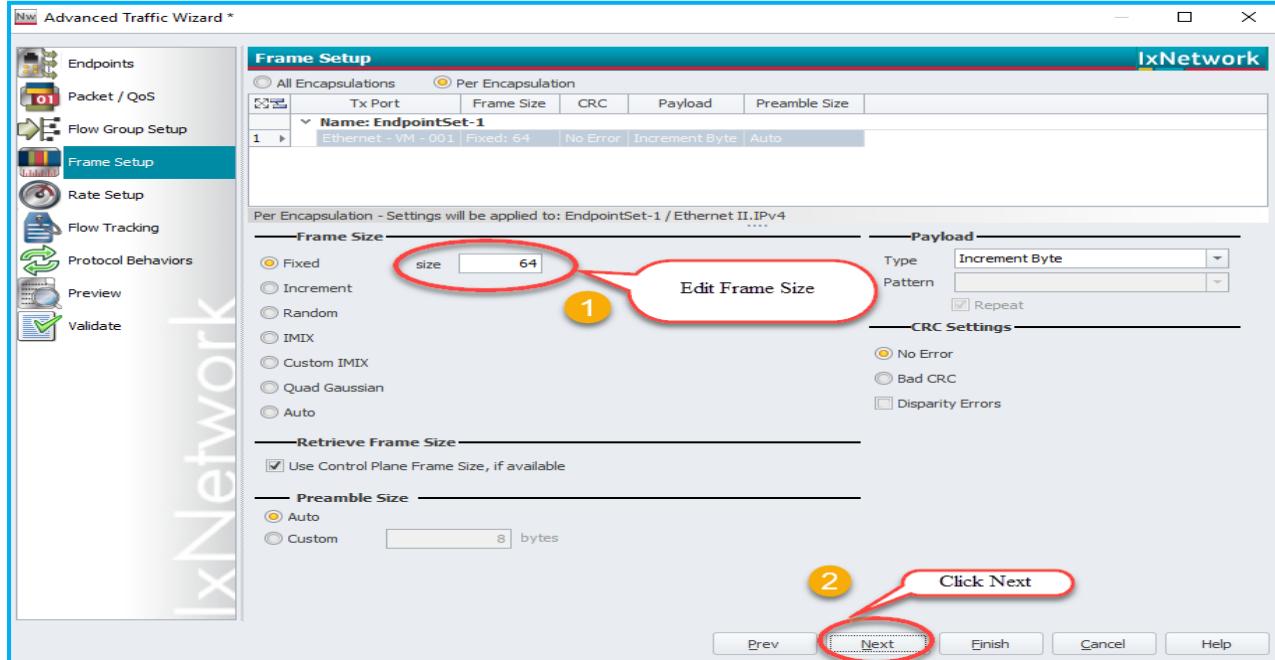
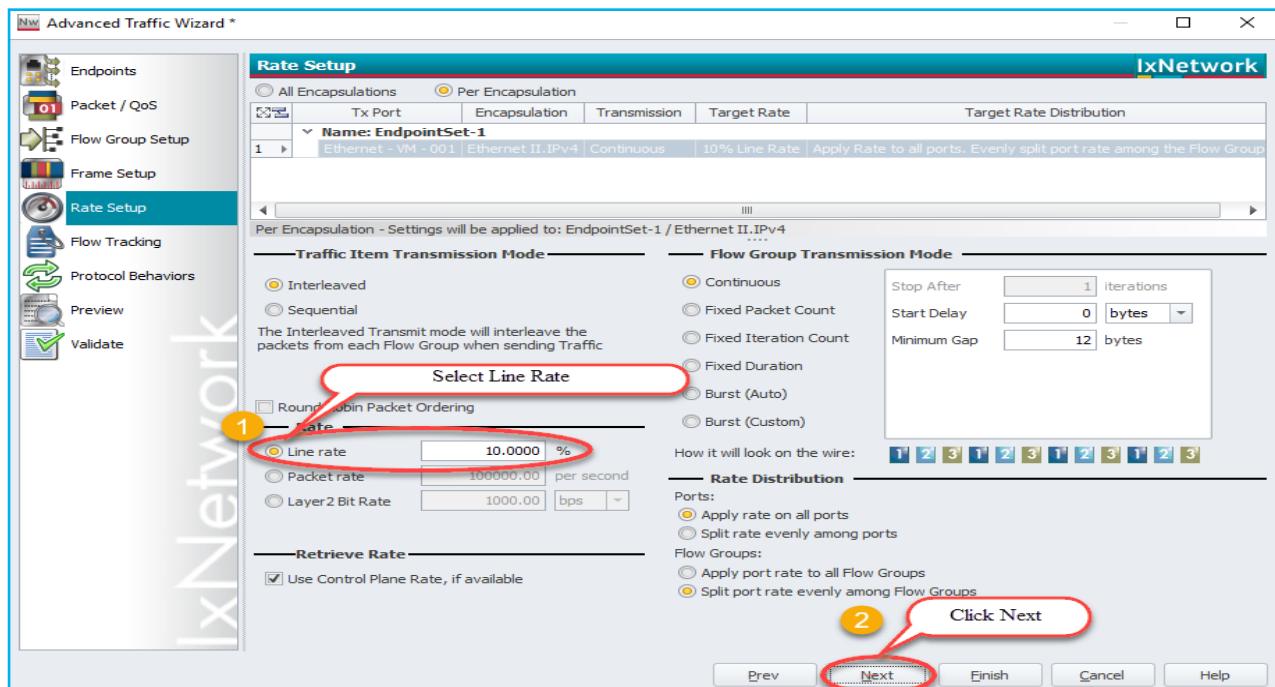


Fig 8.5: Setup Frame size as per test scenario

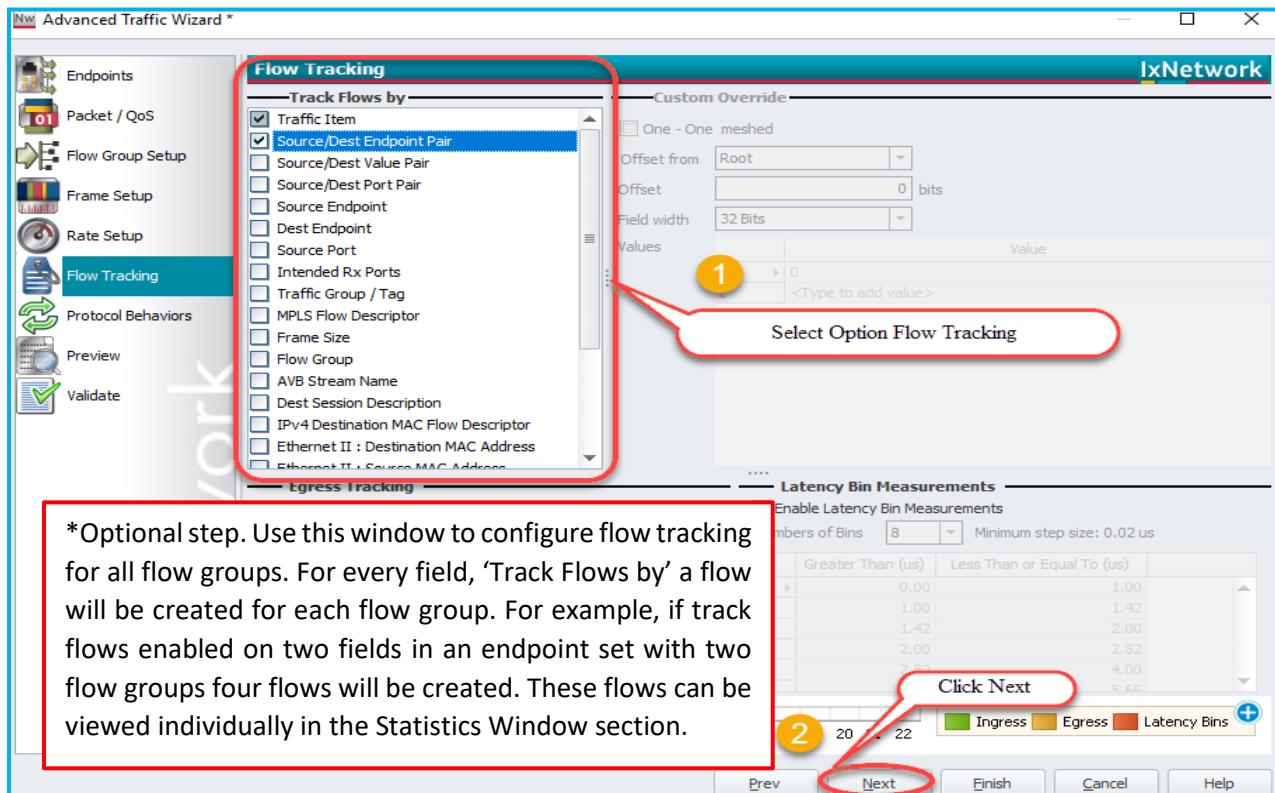
2.13 Setup Line Rate



8.6: Setup line rate

2.14 Setup Flow Tracking

*Setup Flow Tracking and Protocol Behaviors are optional.



*Optional step. Use this window to configure flow tracking for all flow groups. For every field, 'Track Flows by' a flow will be created for each flow group. For example, if track flows enabled on two fields in an endpoint set with two flow groups four flows will be created. These flows can be viewed individually in the Statistics Window section.

Fig 8.7 Setup flow tracking to track specific field

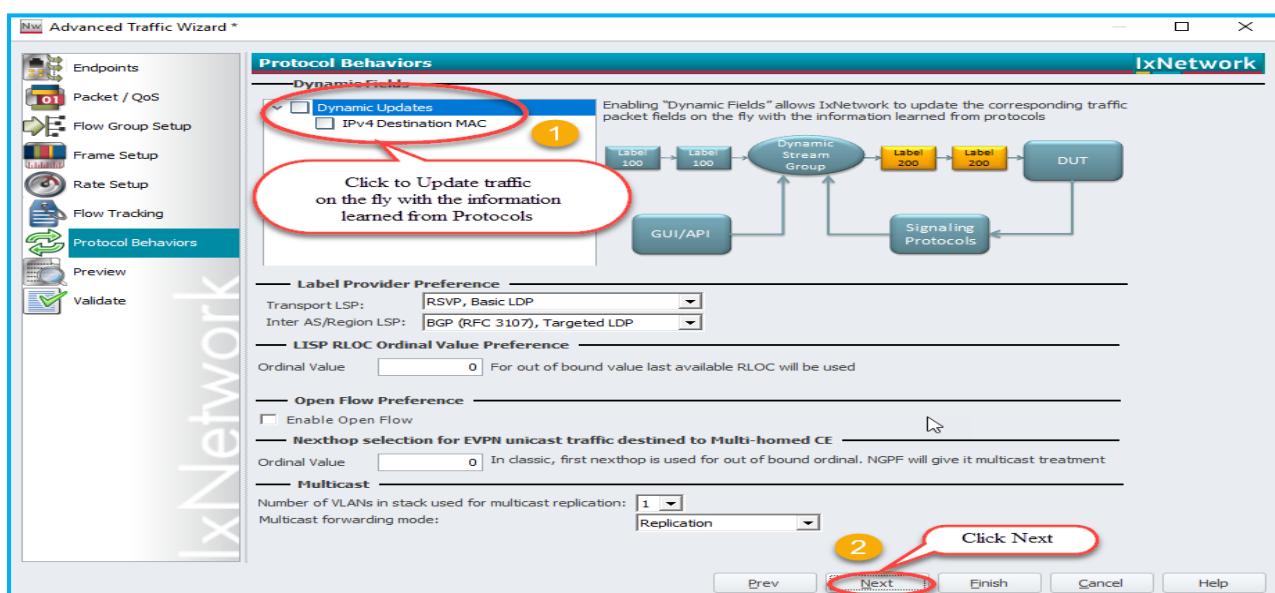


Fig 8.8 Update traffic on the fly with information learned from protocols

2.15 Preview Flow Groups

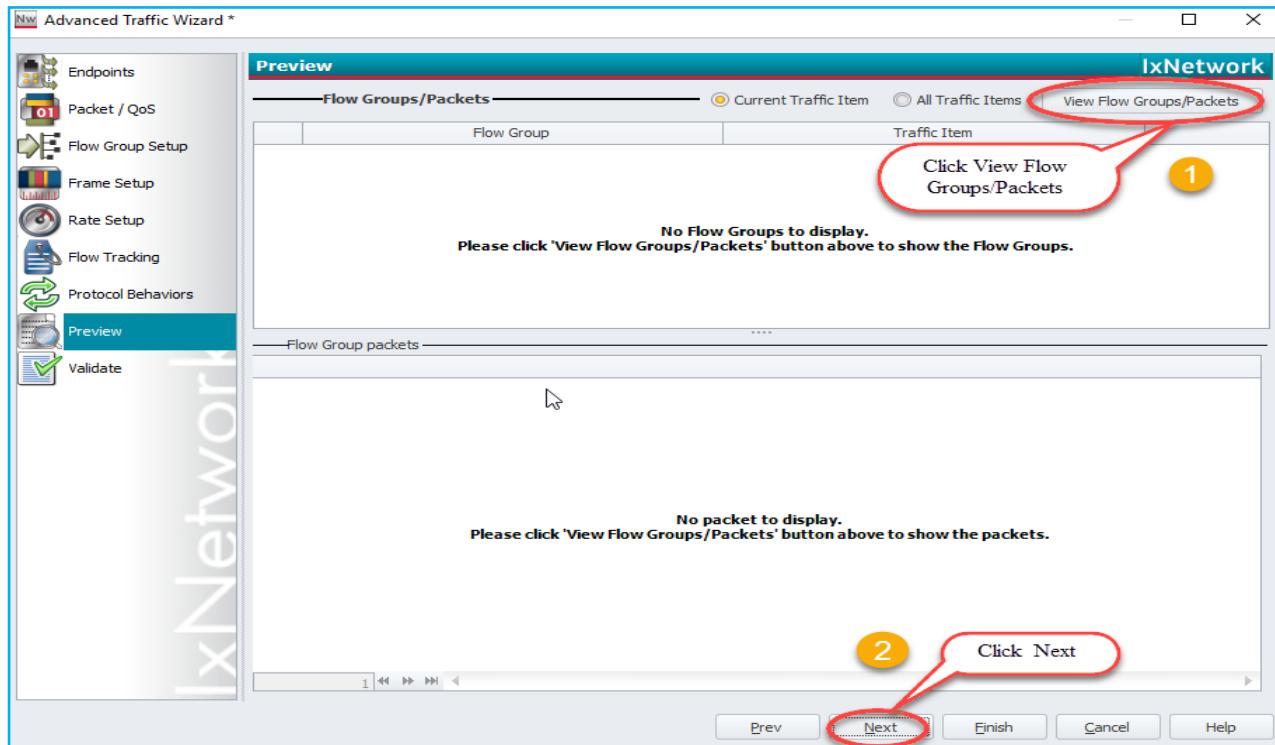


Fig 8.9 View flow group which is currently configured

2.16 Validate Traffic Items

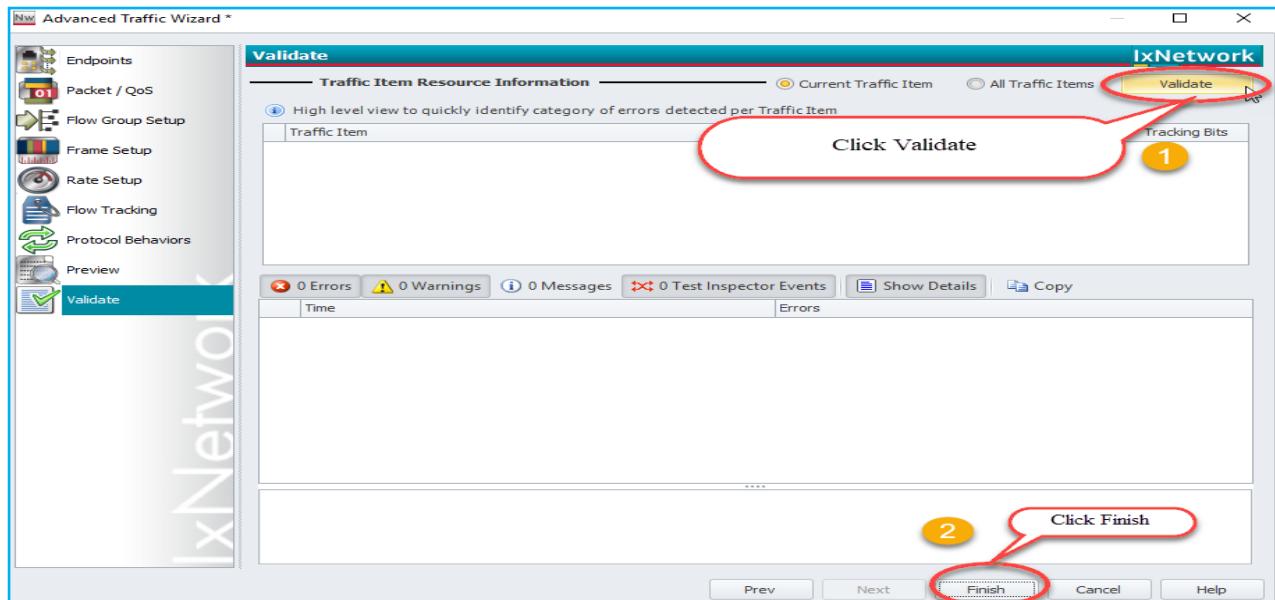


Fig 8.10 Validate the traffic item to identify errors

2.17 Apply Traffic, Start Traffic and Statistics View



Click this **Regenerate** button to regenerate the traffic before apply. If this button clicked, IxNetwork detects conflicts between existing and newly-generated flow groups (If created) within the traffic item and accept the newly-generated flow groups if needed.

Apply the L2-L3 Traffic by selecting **Apply L2-L3 Traffic** from the L2-3Traffic drop-down menu (on the **Traffic Tools** tab) and Start L2-L3 Traffic.

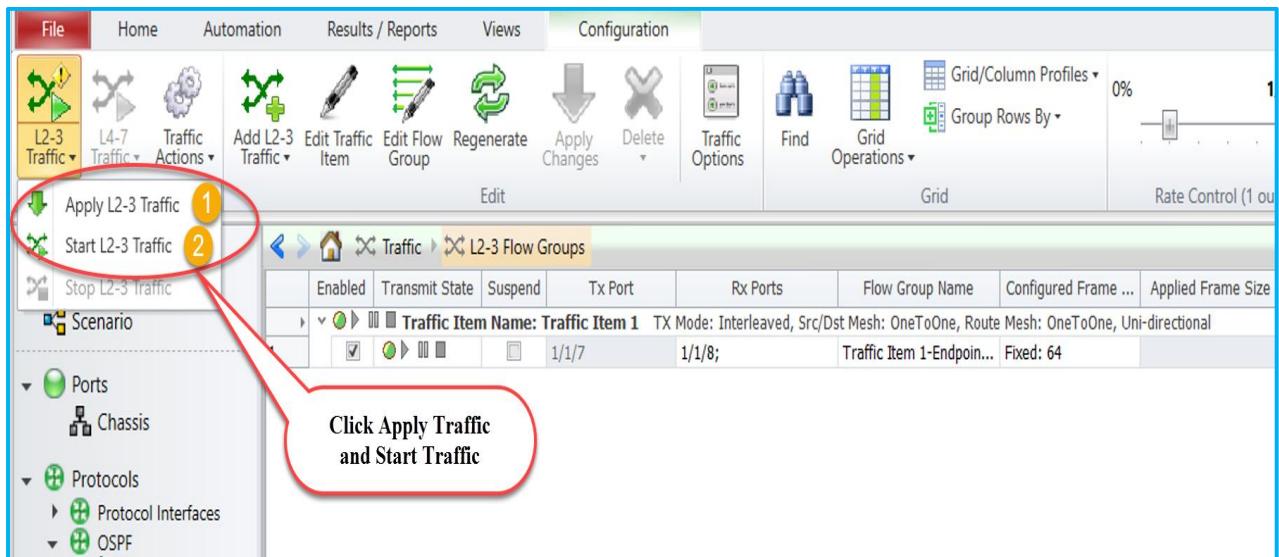


Fig 9.1 Apply and start traffic

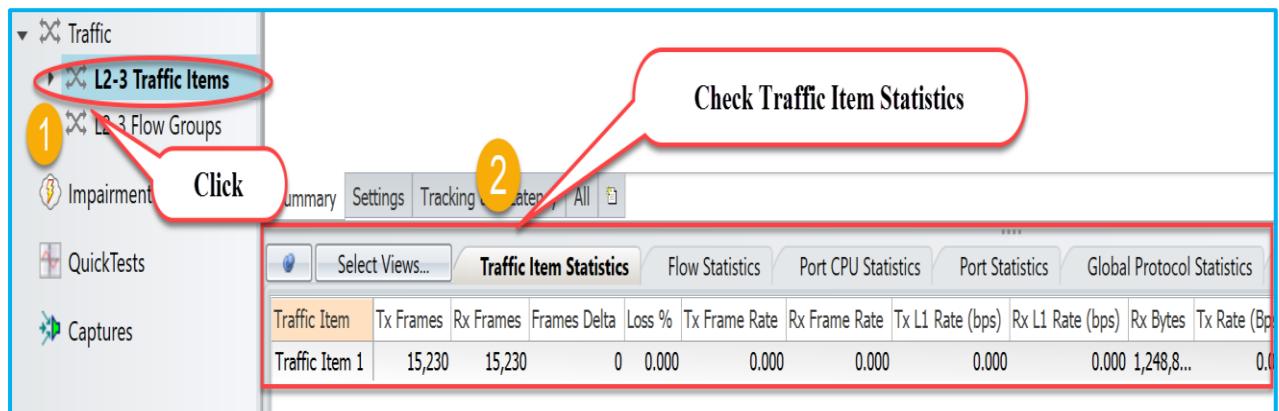


Fig 9.2 Verify traffic statistics

3. Configure OSPFv2 through Automation

This section explains the method to automate OSPFv2 emulation scenario through High Level TCL/Python APIs. Covers common and OSPFv2 specific HLAPIs used in IxNetwork-Classic framework.

3.1 Initialize Environment

Source Ixia package and proceed with HLTAPI execution.

<u>TCL</u>	<u>PYTHON</u>
<pre>> package require Ixia Tcl 8.5 is installed on 64bit architecture. IXIA_VERSION env variable is set to 8.20.0.10, but this value is not matching any HLTSET. Using default HLTSET (HLTSET210) instead. Using products based on HLTSET210 IxTclHal is not be used for current HLTSET. Loaded IxTclNetwork 8.20.1071.8 Loaded Mpexpr 1.0 HLT release 8.20.136.2 Loaded ixia_hl_lib-8.20 8.20 > package require Tclx 8.4</pre>	<pre>import sys, os import time, re from ixiatcl import IxiaTcl from ixiahlt import IxiaHlt from ixiaerror import IxiaError tcl_dependencies = ['/usr/local/lib/', '/usr/lib/', '/usr/share/tcl8.5', '/usr/lib/tcl8.5', '/usr/lib/tk8.5', '/usr/share/tk8.5'] ixiatcl = IxiaTcl(tcl_autompath=tcl_dependencies) ixia = IxiaHlt(ixiatcl, use_legacy_api = 1) Note: If python version > 3.4, call IxiaTcl with Tcl 8.6 path. Example: tcl_dependencies = ['/path/to/tcl8.6']; ixiatcl = IxiaTcl(tcl_autompath=tcl_dependencies) ixiatcl = IxiaTcl(); ixia = IxiaHlt(ixiatcl)</pre>

3.2 Add Chassis and Reserve Ports

::ixia::connect - Connects to the Ixia Chassis, takes ownership of selected ports and optionally loads a configuration on the chassis or resets the targeted ports to factory defaults.

<u>TCL</u>	<u>PYTHON</u>
<pre>> set connect_result [::ixia::connect \ -device 10.39.64.132 \ -port_list {1/7 1/8} \ -reset 1 \ -ixnetwork_tcl_server 10.154.161.223:8009 \ -tcl_server 10.154.161.223] > set ports [keyget connect_result vport_list] > set port1 [lindex \$ports 0] > set port2 [lindex \$ports 1]</pre>	<pre>> connect_result = ixia.connect (device = 10.39.64.132, port_list = "1/7 1/8", reset = 1 ixnetwork_tcl_server = "10.154.161.223:8009", tcl_server = 10.154.161.223) > ports = connect_result['vport_list'].split() > port1 = ports[0] > port2 = ports[1]</pre>

*Note: High Level API's are highlighted in Red and all other handles are highlighted in Green.

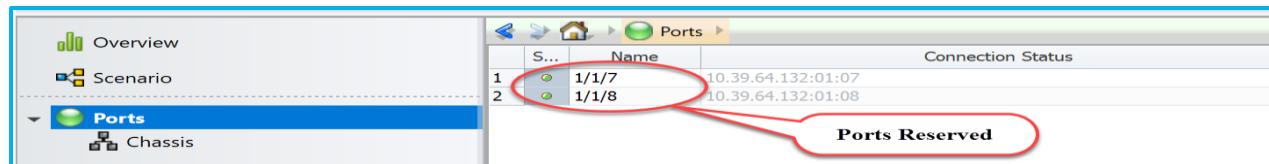


Fig 10.1 Connected to chassis and reserved ports

3.3 Configure Ports

::ixia::interface_config - Configures an interface and accommodates addressing schemes such as IPv4, IPv6, MAC and VLAN.

<u>TCL</u>	<u>PYTHON</u>
<pre>> set intf_cfg_stats1 [::ixia::interface_config -port_handle \$port1 -intf_ip_addr 198.168.1.2 -gateway 198.168.1.1 -arp_send_req 1 -autonegotiation 1 -phy_mode fiber]</pre>	<pre>> intf_cfg_stats1 = ixia.interface_config (port_handle = port1, intf_ip_addr = 198.168.1.2, gateway = 198.168.1.1, arp_send_req = 1, autonegotiation = 1, phy_mode = 'fiber')</pre>
<pre>> set intf_cfg_stats2 [::ixia::interface_config -port_handle \$port2 -intf_ip_addr 198.168.1.1 -gateway 198.168.1.2 -arp_send_req 1 -autonegotiation 1 -phy_mode fiber]</pre>	<pre>> intf_cfg_stats2 = ixia.interface_config (port_handle = port2, intf_ip_addr = 198.168.1.1, gateway = 198.168.1.2, arp_send_req = 1, autonegotiation = 1, phy_mode = 'fiber')</pre>

Fig 11.1 Configure ports attributes using HLAPI

The screenshot shows the IXIA LabVIEW software interface. On the left, there's a navigation tree with options like Overview, Scenario, Ports, Protocols, Traffic, Impairments, QuickTests, and Captures. Under Protocols, 'Protocol Interfaces' is selected. The main area has tabs for Connected Interfaces, Unconnected Interfaces, SFE Tuner, Discovered Neighbors, Interface Addresses, DHCPv4 Discovered Information, DHCPv6 Discovered Information, and Rate Control Parameters. In the 'Connected Interfaces' tab, there are several checkboxes at the top: ARP on Link Up, Send Single ARP per Gateway, NS on Link Up, Send Single NS per Gateway, Filter By Unresolved Interfaces, and Send NS To Neighbor If Gateway Unavailable. Below these are two rows of interface configuration tables. The first row shows an interface with Port Description '1/1/7 - 100/1000 Base X', Port Link '1/1/7 - 00 00 00 00 00 04 -', Interface Description '1/1/7 - 00 00 00 00 00 04 -', IPv4 Address '198.168.1.2', IPv4 Mask Width '24', and Gateway '198.168.1.1'. The second row shows an interface with Port Description '1/1/8 - 100/1000 Base X', Port Link '1/1/8 - 00 00 00 00 00 05 -', Interface Description '1/1/8 - 00 00 00 00 00 05 -', IPv4 Address '198.168.1.1', IPv4 Mask Width '24', and Gateway '198.168.1.2'. A red circle highlights the second row. A red callout bubble points to the second row with the text 'Configured Ports Attributes'.

	Port Description	Port Link	Interface Description	IPv4 Address	IPv4 Mask Width	Gateway
1	1/1/7 - 100/1000 Base X	1/1/7 - 00 00 00 00 00 04 -	1/1/7 - 00 00 00 00 00 04 -	198.168.1.2	24	198.168.1.1
2	1/1/8 - 100/1000 Base X	1/1/8 - 00 00 00 00 00 05 -	1/1/8 - 00 00 00 00 00 05 -	198.168.1.1	24	198.168.1.2

Fig 11.2 View configured attributes in GUI

3.4 Create OSPFv2

`::ixia::emulation_ospf_config` - Add ospf router(s) to a port and configure OSPFv2 attributes. Disable 'discard learned lsa' to view learned lsa's on peer router.

<u>TCL</u>	<u>PYTHON</u>
<pre>> set ospf_emul_res1 \ [::ixia::emulation_ospf_config -port_handle \$port1 -mode create -session_type ospfv2 -area_id 0.0.0.1 -router_id 190.29.0.1 -network_type ptop -lsa_discard_mode 0 -intf_ip_addr 198.168.1.2 -neighbor_intf_ip_addr 192.168.1.1] > set ospf_handle1 [keyget ospf_emul_res1 \ handle]</pre>	<pre>> ospf_emul_res1 = \ ixia.emulation_ospf_config (port_handle = port1, mode = create, session_type = 'ospfv2', area_id = 0.0.0.1, router_id = 190.29.0.1, network_type = 'ptop', lsa_discard_mode = 0, intf_ip_addr = 198.168.1.2, neighbor_intf_ip_addr = 192.168.1.1) > ospf_handle1 = ospf_emul_res1['handle']</pre>
<pre>> set ospf_emul_res2 \ [::ixia::emulation_ospf_config -port_handle \$port2 -mode create -session_type ospfv2 -area_id 0.0.0.1 -router_id 190.30.0.1 -network_type ptop -lsa_discard_mode 0 -intf_ip_addr 198.168.1.1 -neighbor_intf_ip_addr 192.168.1.2] > set ospf_handle2 [keyget ospf_emul_res2 \ handle]</pre>	<pre>> ospf_emul_res2 = \ ixia.emulation_ospf_config (port_handle = port2, mode = create, session_type = 'ospfv2', area_id = 0.0.0.1, router_id = 190.30.0.1, network_type = 'ptop', lsa_discard_mode = 0, intf_ip_addr = 198.168.1.1, neighbor_intf_ip_addr = 192.168.1.2) > ospf_handle2 = ospf_emul_res2['handle']</pre>

Fig 12.1 Configure OSPFv2 attributes using HLA API

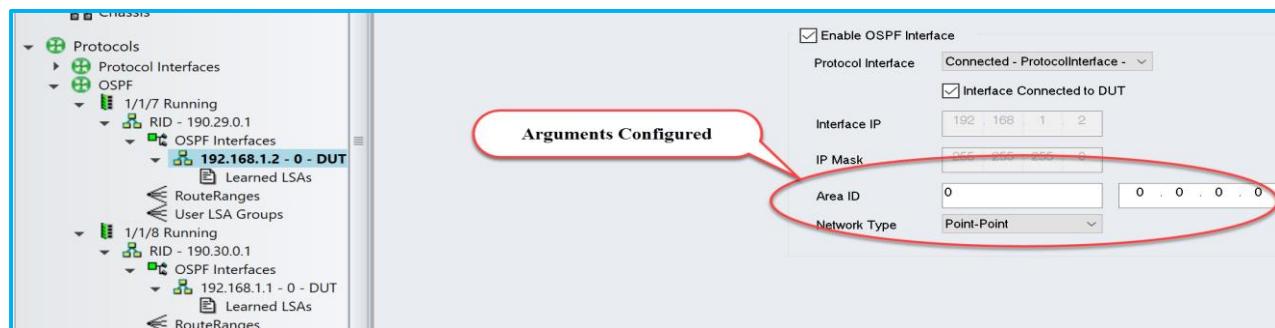


Fig 12.2 View configured ospfv2 attributes on 1/1/7 in GUI

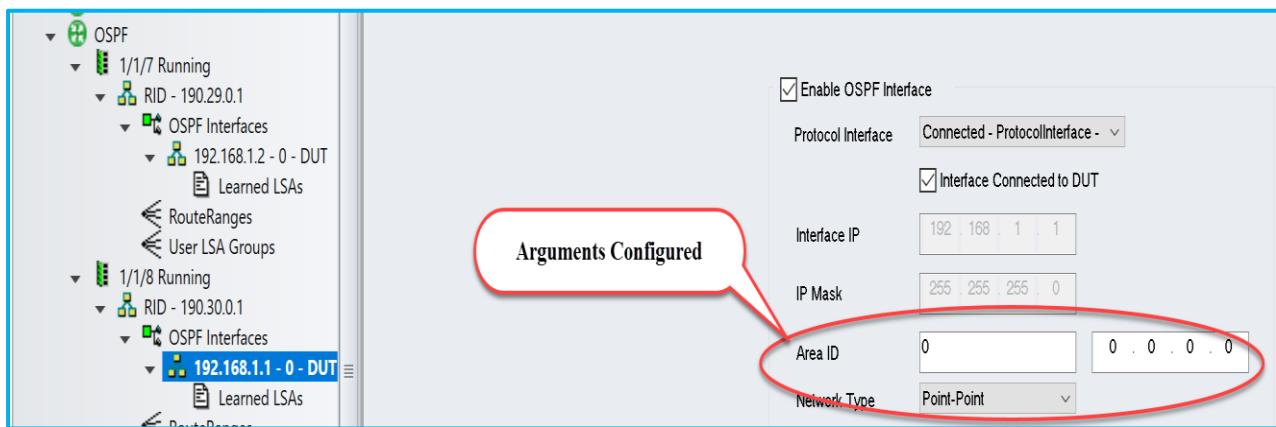


Fig 12.3 View configured ospfv3 attributes on 1/1/8 in GUI

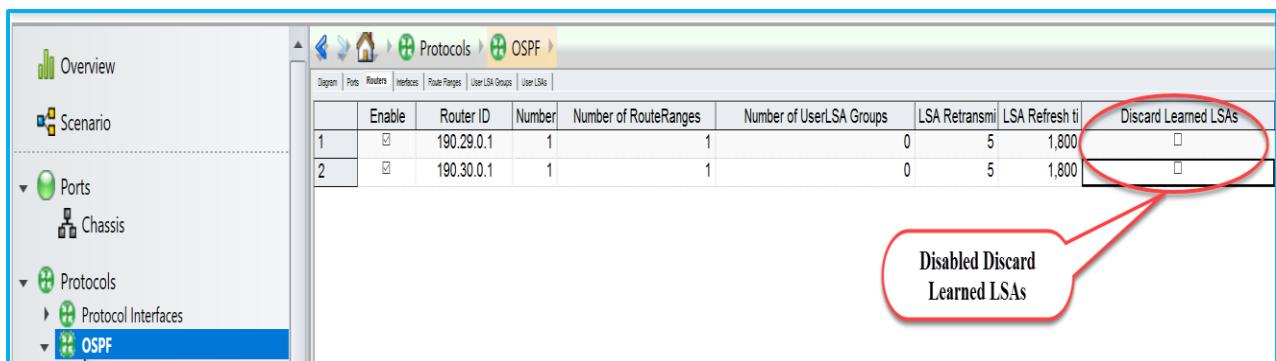


Fig 12.4 View Discard Learned LSAs disabled in GUI

3.5 Create OSPFv2 Route Ranges

`::ixia::emulation_ospf_topology_route_config` - Add OSPFv2 route(s) to a simulated OSPFv2 router interface.

<u>TCL</u>	<u>PYTHON</u>
<pre>> set ospf_routes \ [:ixia::emulation_ospf_topology_route_config \ -mode create \ -handle \$ospf_handle1 \ -type ext_routes \ -external_number_of_prefix 1 \ -external_prefix_length 24 \ -external_prefix_start 5.1.1.0 \ -external_prefix_step 1]</pre>	<pre>> ospf_routes = \ ixia.emulation_ospf_topology_route_config(\ mode = 'create', \ handle = ospf_handle1, \ type = ext_routes, \ external_number_of_prefix = 1, \ external_prefix_length = 24, \ external_prefix_start = 5.1.1.0, \ external_prefix_step = 1)</pre>

Fig 13.1 Configure OSPFv2 Route Ranges on 1/1/7 ospf handle using HLAPI

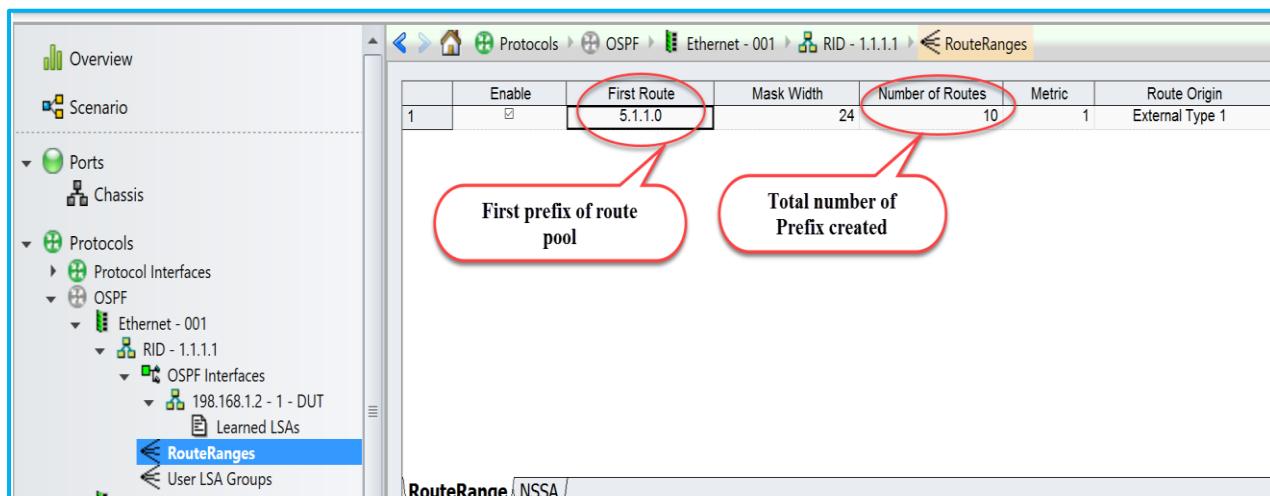


Fig 13.2 View configured route ranges in 1/1/7 (Ethernet – 001)

3.6 Start Protocols

::ixia::emulation_ospf_control - Start OSPF protocol.

<u>TCL</u>	<u>PYTHON</u>
> set result1 [::ixia::emulation_ospf_control -handl \$ospf_handle1 -port_handle \$port1 -mode start]	> result1 = ixia.emulation_ospf_control (handle = ospf_handle1, port_handle = port1, mode = 'start')
> set result2 [::ixia::emulation_ospf_control -handl \$ospf_handle2 -port_handle \$port2 -mode start]	> result2 = ixia.emulation_ospf_control (handle = ospf_handle2, port_handle = port2, mode = 'start')
> puts "Wait for some time for the protocols to converge"	> print """Wait for some time for the protocols to converge"""

Fig 14.1 Start OSPFv2 protocols using HAPI

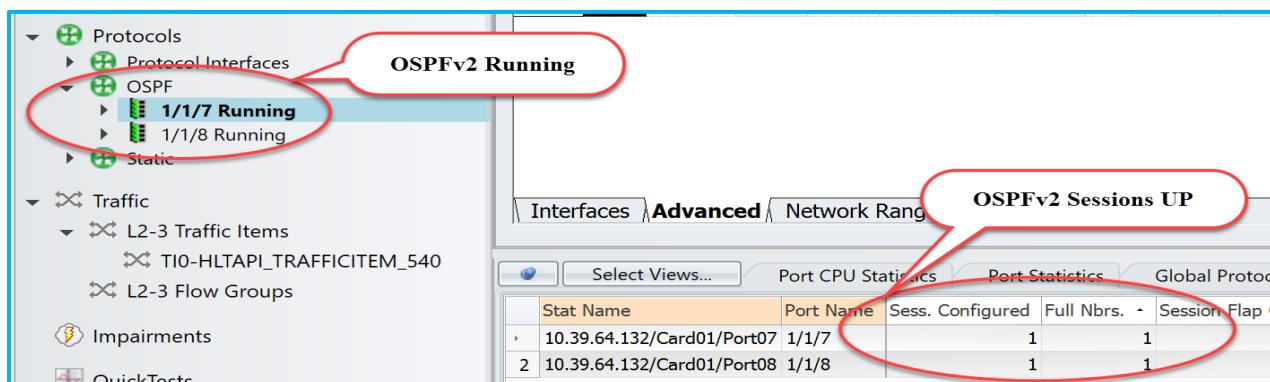


Fig 14.2 View ospfv2 session states details in GUI

3.7 Check Learned LSAs

::ixia::emulation_ospf_info - Retrieves information about the OSPF sessions. Please refer Fig 15.2

TCL

```
> set learned_info [::ixia::emulation_ospf_info
    -mode learned_info
    -handle $ospf_handle2] \\
> set sessions [keylget learned_info
$ospf_handle2.$$ospf_handle2/interface:1]
> puts "Learned LSAs: $sessions"
1:
  adv_router_id: 190.29.0.1
  age: 30
  link_state_id: 190.29.0.1
  lsa_type: router
  seq_number: 80000002
  prefix_v4_address: ::ixNet::OK
  prefix_v6_address: ::ixNet::OK
  prefix_length: ::ixNet::OK
2:
  adv_router_id: 190.29.0.1
  age: 41
  link_state_id: 5.1.1.0
  lsa_type: external
  seq_number: 80000001
  prefix_v4_address: ::ixNet::OK
  prefix_v6_address: ::ixNet::OK
  prefix_length: ::ixNet::OK
3:
  adv_router_id: 190.29.0.1
  age: 41
  link_state_id: 5.1.2.0
  lsa_type: external
  seq_number: 80000001
  prefix_v4_address: ::ixNet::OK
----- Stripped off Long output -----
9:
  adv_router_id: 190.29.0.1
  age: 41
  link_state_id: 5.1.8.0
  lsa_type: external
  seq_number: 80000001
  prefix_v4_address: ::ixNet::OK
  prefix_v6_address: ::ixNet::OK
  prefix_length: ::ixNet::OK
```

PYTHON

```
> learned_info = ixia.emulation_ospf_info(
    mode = 'learned_info',
    handle = ospf_handle2) \\
> sessions =
learned_info[ospf_handle2][ospf_handle2+'
/interface:1']
> print "Learned LSAs: %s" % sessions
1:
  adv_router_id: 190.29.0.1
  age: 30
  link_state_id: 190.29.0.1
  lsa_type: router
  seq_number: 80000002
  prefix_v4_address: ::ixNet::OK
  prefix_v6_address: ::ixNet::OK
  prefix_length: ::ixNet::OK
2:
  adv_router_id: 190.29.0.1
  age: 41
  link_state_id: 5.1.1.0
  lsa_type: external
  seq_number: 80000001
  prefix_v4_address: ::ixNet::OK
  prefix_v6_address: ::ixNet::OK
  prefix_length: ::ixNet::OK
3:
  adv_router_id: 190.29.0.1
  age: 41
  link_state_id: 5.1.2.0
  lsa_type: external
  seq_number: 80000001
  prefix_v4_address: ::ixNet::OK
----- Stripped off Long output -----
9:
  adv_router_id: 190.29.0.1
  age: 41
  link_state_id: 5.1.8.0
  lsa_type: external
  seq_number: 80000001
  prefix_v4_address: ::ixNet::OK
  prefix_v6_address: ::ixNet::OK
  prefix_length: ::ixNet::OK
```

<pre> 10: adv_router_id: 190.29.0.1 age: 41 link_state_id: 5.1.9.0 lsa_type: external seq_number: 80000001 prefix_v4_address: ::ixNet::OK prefix_v6_address: ::ixNet::OK prefix_length: ::ixNet::OK 11: adv_router_id: 190.29.0.1 age: 41 link_state_id: 5.1.10.0 lsa_type: external seq_number: 80000001 prefix_v4_address: ::ixNet::OK prefix_v6_address: ::ixNet::OK prefix_length: ::ixNet::OK status: 1 log: </pre>	<pre> 10: adv_router_id: 190.29.0.1 age: 41 link_state_id: 5.1.9.0 lsa_type: external seq_number: 80000001 prefix_v4_address: ::ixNet::OK prefix_v6_address: ::ixNet::OK prefix_length: ::ixNet::OK 11: adv_router_id: 190.29.0.1 age: 41 link_state_id: 5.1.10.0 lsa_type: external seq_number: 80000001 prefix_v4_address: ::ixNet::OK prefix_v6_address: ::ixNet::OK prefix_length: ::ixNet::OK status: 1 log: </pre>
--	--

Fig 15.1 Retrieve learned LSA info using HLAPI

The screenshot shows the IXIA software interface with the following details:

- Left Panel (Tree View):**
 - Overview
 - Scenario
 - Ports
 - Chassis
 - Protocols
 - Protocol Interfaces
 - OSPF
 - 1/1/7 Running
 - RID - 190.29.0.1
 - OSPF Interfaces
 - RouteRanges
 - User LSA Groups
 - 1/1/8 Running
 - RID - 190.30.0.1
 - OSPF Interfaces
 - 192.168.1.1 - 0 - DUT
 - Learned LSAs
 - RouteRanges
 - User LSA Groups
- Right Panel (Table View):**

Number of LSAs 11

	Link State ID	Advertising Router	LSA Type
1	190.29.0.1	190.29.0.1	Router
2	5.1.1.0	190.29.0.1	Summary IP
3	5.1.2.0	190.29.0.1	Summary IP
4	5.1.3.0	190.29.0.1	Summary IP
5	5.1.4.0	190.29.0.1	Summary IP
6	5.1.5.0	190.29.0.1	Summary IP
7	5.1.6.0	190.29.0.1	Summary IP
8	5.1.7.0	190.29.0.1	Summary IP
9	5.1.8.0	190.29.0.1	Summary IP
10	5.1.9.0	190.29.0.1	Summary IP
11	5.1.10.0	190.29.0.1	Summary IP
- Bottom Panel (Buttons):**
 - Select Views...
 - Port CPU Statistics
 - Port Statistics
 - Stat Name Port Name Sess. Configured Full Nbr

A red box highlights the "Learned LSAs" section in the left tree view, and a red callout bubble points to the "Learned LSAs from Peer" entry in the bottom panel.

Fig 15.2: View learned LSAs info in GUI

3.8 Configure Traffic

::ixia::traffic_config - Configures traffic streams on the specified ports with the specified options.
Here created the traffic flow from 1/1/8 -> 1/1/7.

TCL	PYTHON
<pre>> set traffic_res [::ixia::traffic_config -mode create -transmit_mode continuous -track_by {traffic_item} -rate_pps 1000 -port_handle \$port2 -port_handle2 \$port1 -l3_protocol ipv4 -ip_src_addr 192.168.1.1 -ip_dst_addr 192.168.1.2 -mac_dst 00:00:19:d5:54:74 -mac_src 00:00:19:d5:54:75 -l3_length 64]</pre>	<pre>> traffic_res = ixia.traffic_config (mode = 'create', transmit_mode = 'continuous', track_by = 'traffic_item', rate_pps = 1000, port_handle = port2, port_handle2 = port1, l3_protocol = 'ipv4', ip_src_addr = 192.168.1.1, ip_dst_addr = 192.168.1.2, mac_dst = 00:00:19:d5:54:74, mac_src = 00:00:19:d5:54:75, l3_length = 64)</pre>

Fig 16.1: Create traffic stream from 1/1/8 to 1/1/7 port handle using HLAPI

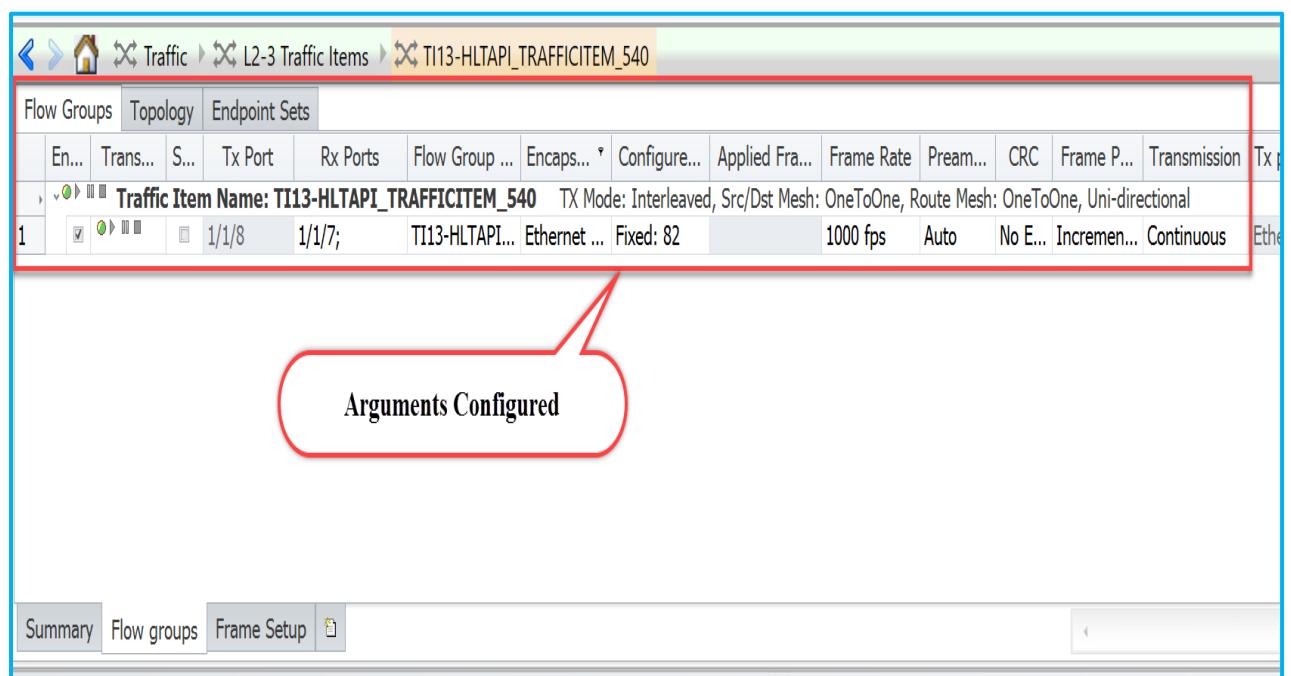


Fig 16.2: View traffic item configuration in GUI

3.9 Start and Stop Traffic

::ixia::traffic_control - Starts or Stops traffic on a given port list. As per below code, start the traffic, wait for some time for the traffic to flow and stop the traffic.

<u>TCL</u>	<u>PYTHON</u>
> ::ixia::traffic_control -action run	> ixia.traffic_control (action = 'run')
> puts "Wait for some time for the traffic to flow"	> print "Wait for some time for the traffic to flow"
> ::ixia::traffic_control -action stop	> ixia.traffic_control (action = 'stop')

Fig 17.1: Start and stop the traffic using HLAPI

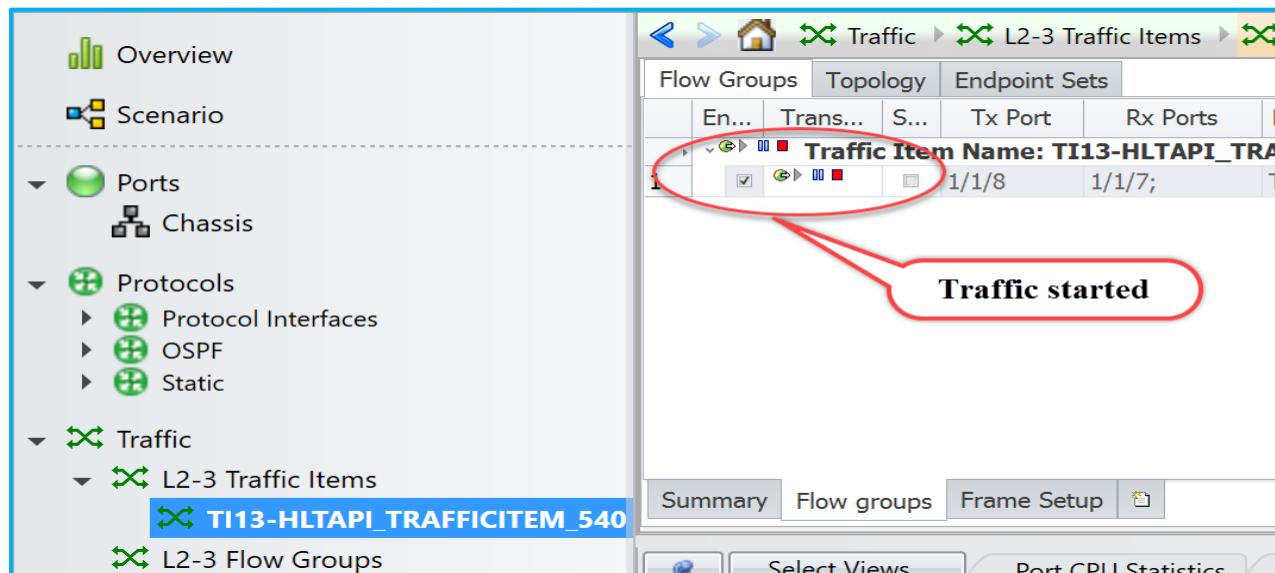


Fig 17.2: View traffic start status in GUI

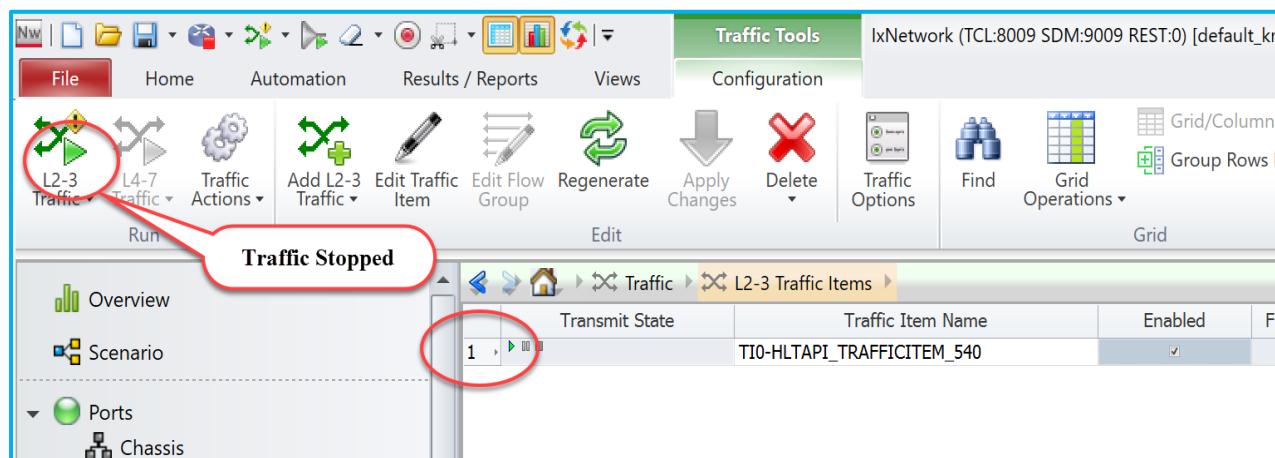


Fig 17.3: View traffic stop status in GUI

3.10 Get Statistics

::ixia::traffic_stats - Gathers statistics depends on the mode suchss as per_port_flows, session, stream, streams, all etc., Below code used mods as 'streams'

<u>TCL</u>	<u>PYTHON</u>
<pre>> set stats ::ixia::traffic_stats -mode streams -streams TI13-HLTAPI_TRAFFICITEM_540] > puts "Traffic Stats: \$stats" status: 1 measure_mode: mixed waiting_for_stats: 0 1/1/7: stream: TI13-HLTAPI_TRAFFICITEM_540: rx: total_pkt_rate: 0.000 total_pkt_byte_rate: 0.000 loss_percent: 0.000 small_error: N/A total_pkts_bytes: 150153856 expected_pkts: N/A pkt_loss_duration: N/A last_tstamp: 00:00:16.334 total_pkts: 368497 reverse_error: N/A ----- Stripped off long output ----- 1/1/8: stream: TI13-HLTAPI_TRAFFICITEM_540: tx: total_pkts: 368497 total_pkt_rate: 0.000</pre>	<pre>> stats = ixia.traffic_stats(mode = streams, streams = TI13-HLTAPI_TRAFFICITEM_540) > print "Traffic stats: %s" % stats status: 1 measure_mode: mixed waiting_for_stats: 0 1/1/7: stream: TI13-HLTAPI_TRAFFICITEM_540: rx: total_pkt_rate: 0.000 total_pkt_byte_rate: 0.000 loss_percent: 0.000 small_error: N/A total_pkts_bytes: 150153856 expected_pkts: N/A pkt_loss_duration: N/A last_tstamp: 00:00:16.334 total_pkts: 368497 reverse_error: N/A ----- Stripped off long output ----- 1/1/8: stream: TI13-HLTAPI_TRAFFICITEM_540: tx: total_pkts: 368497 total_pkt_rate: 0.000</pre>

Fig 18.1: Retrieve traffic stats using HLAPI

Protocols		Summary												
		Protocol Interfaces		OSPF		Static		Traffic		L2-3 Traffic Items		Traffic Item Statistics		
		Select Views...		Port CPU Statistics		Port Statistics		Global Protocol Statistics		L2-L3 Test Summary Statistics		Flow Statistics		
		Traffic Item	Tx Frames	Rx Frames	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate	Tx L1 Rate (bps)	Rx L1 Rate (bps)	Rx Bytes	Tx Rate (Bps)	Rx Rate (Bps)	Tx R
		TI13-HLTAPI_TRAFFICITEM_540	368,497	368,497	0	0.000	0.000	0.000	0.000	0.000	30,216,754	0.000	0.000	

Fig 18.3: Verify Tx Frames and Rx Frames in GUI

4. Other Utilities

4.1 IxNetwork API Documentation Browser

- The main feature of this application is the ability to browse the API meta data in a hierarchical format. Access each level of the hierarchy with a view of siblings, attributes, execs, errors, and children by clicking on BROWSE.

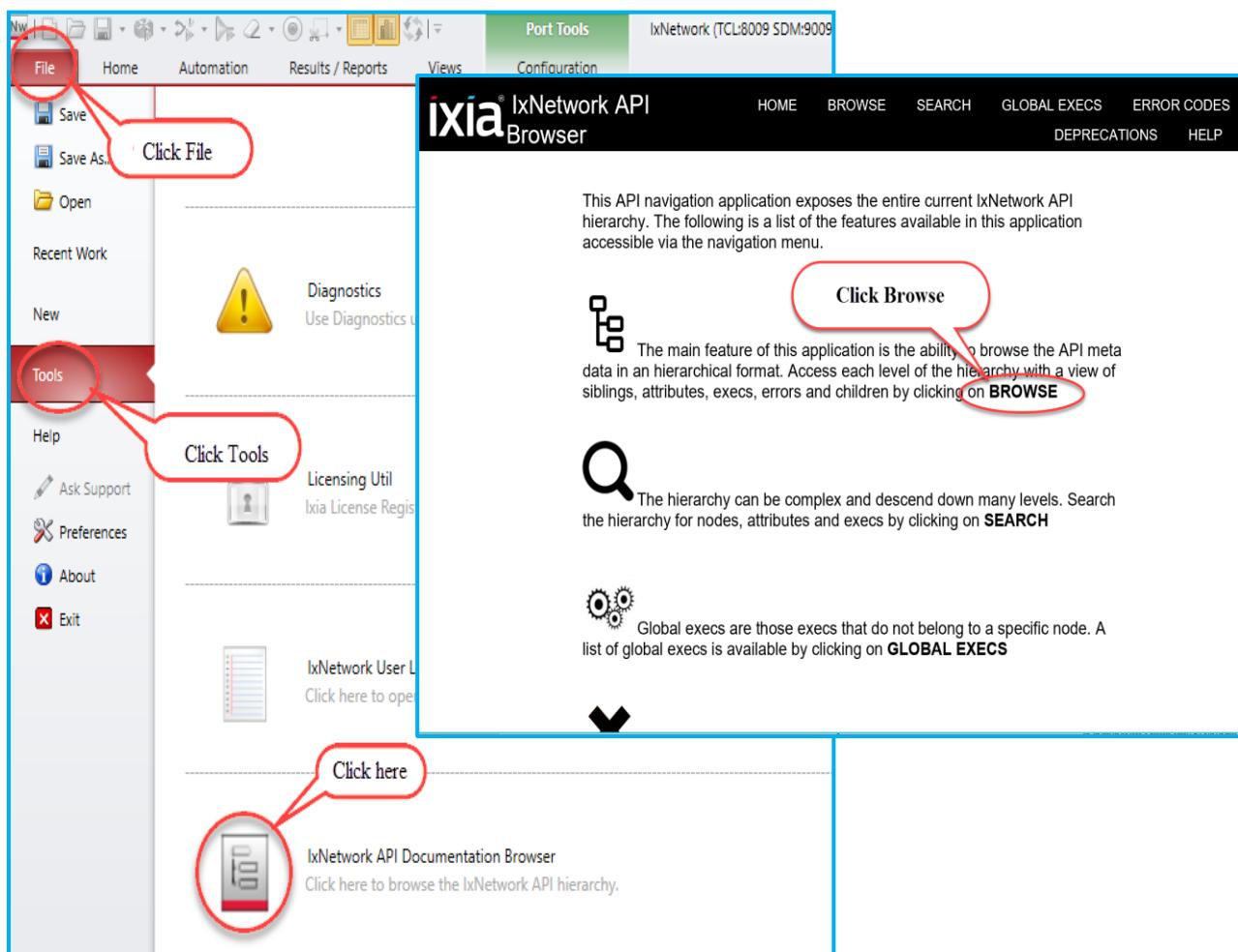


Fig 19.1: IxNetwork API Documentation Browser

4.2 Script Gen

- For complex configuration use SCRIPTGEN. Reverse-engineer the scriptgen scripts as per the requirement.

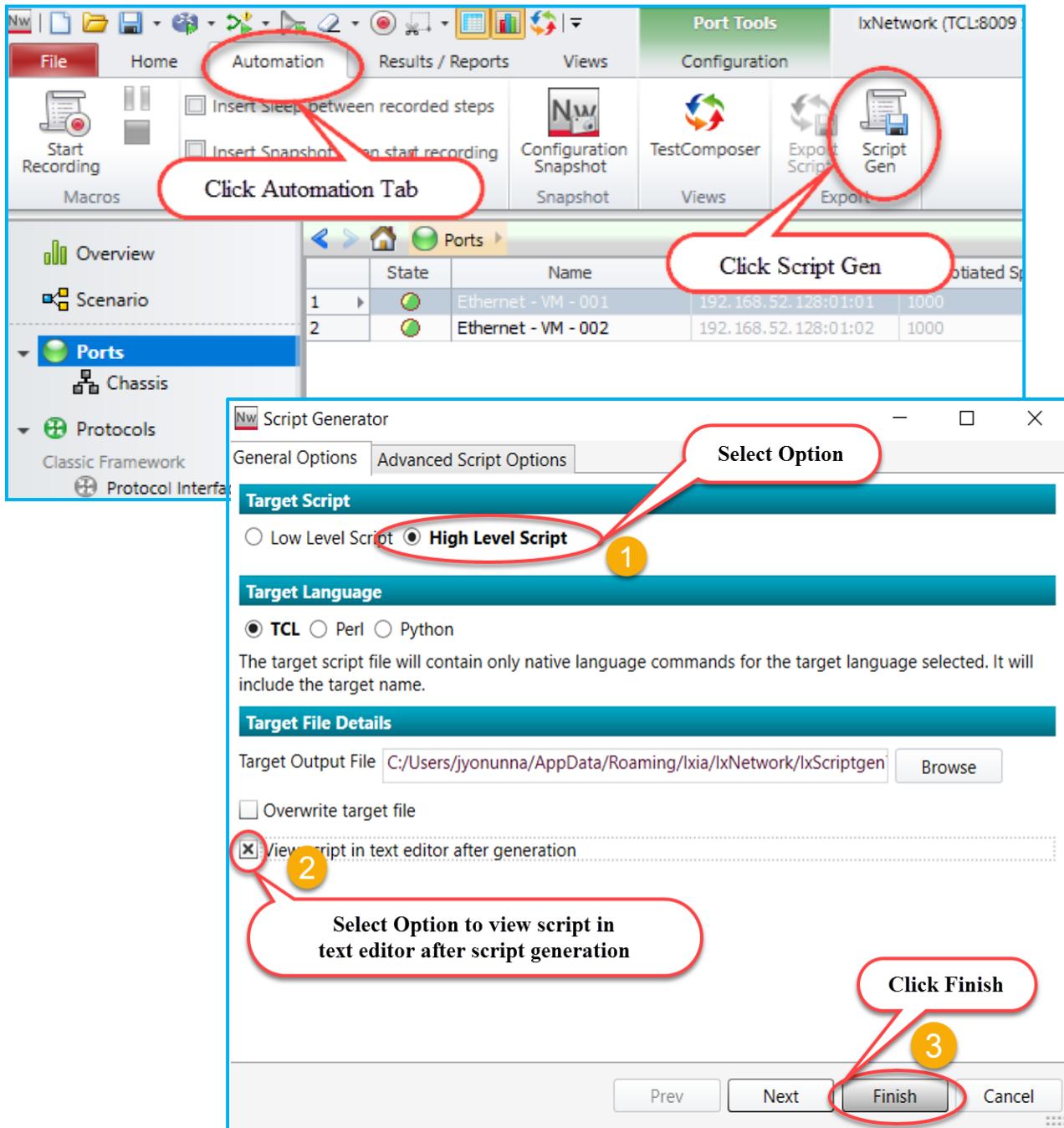


Fig 20.1: Tool to generate script gen

4.3 F1 Option

- Move the mouse pointer over any field in the GUI, and then press F1 to get more information about the field. From Classic Protocols section, users can explore all the fields of all protocols. OSPFv2 protocol tree structure shown below.

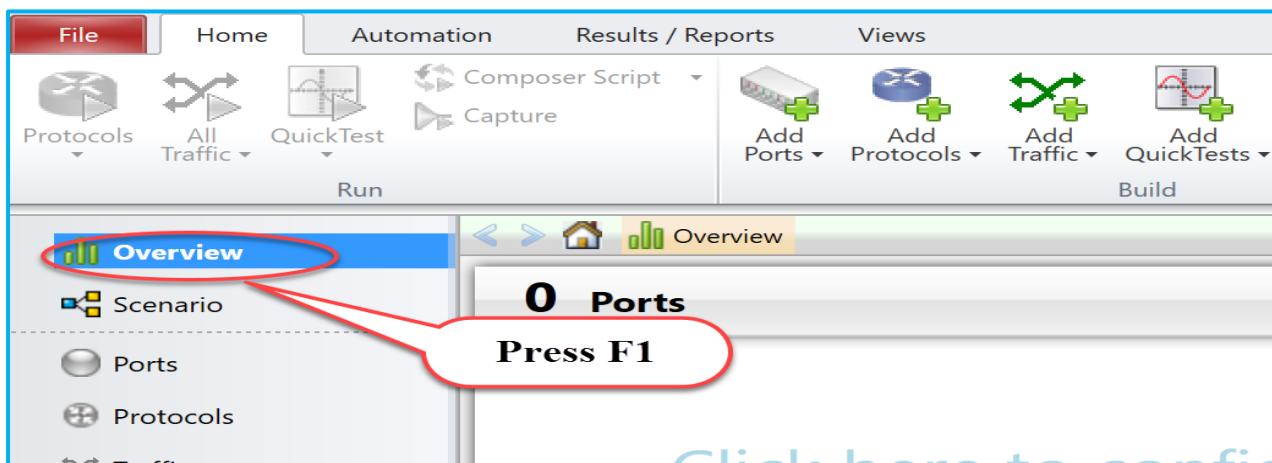


Fig 20.1 Press F1 on Overview field

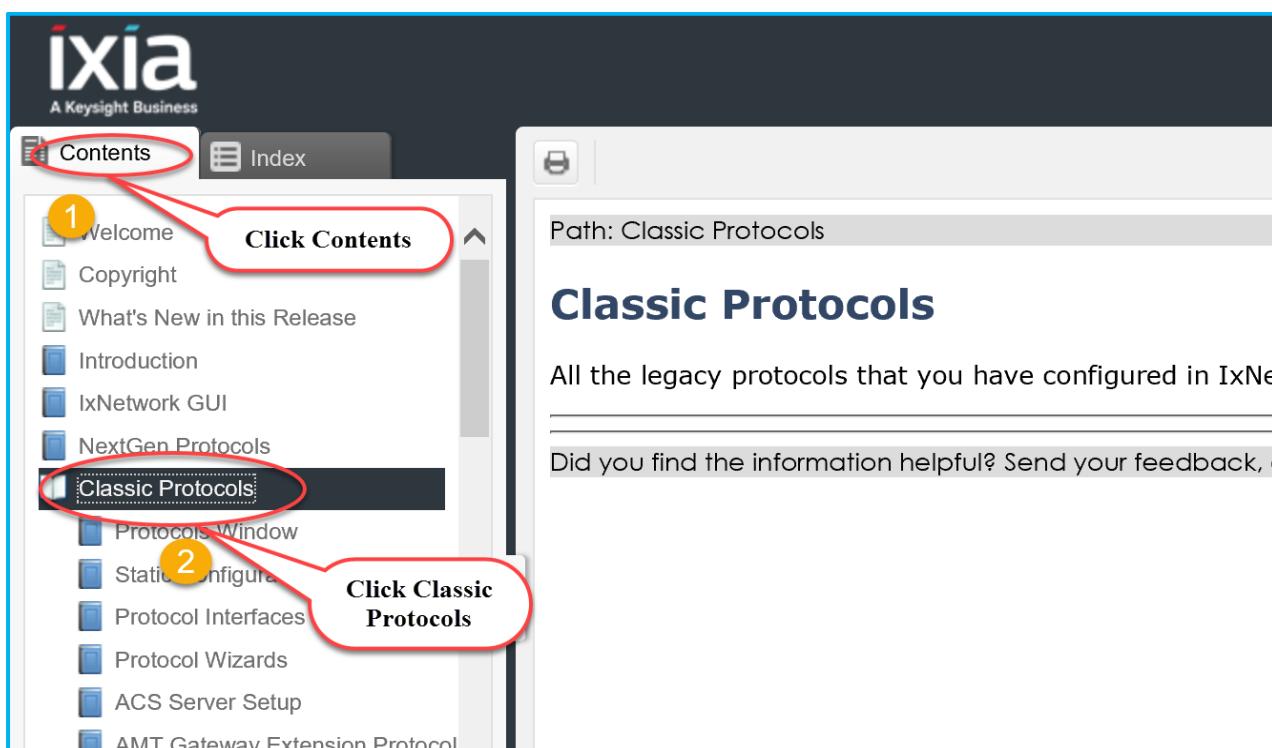


Fig 20.2 Help page to explain about each field in GUI

1 Click OSPF

2 Click OSPFv2 Protocol Tree

3 OSPFv2 Protocol

Protocol Level-Topology diagram & Multiport windows
Port Level-List of routers for this port
Router Level-List of Interfaces for this router
Interfaces Level-List of Interfaces for this router
Interface Level-For an Interface connected to the DUT
Learned LSAs Level-List of learned LSAs for this interface (connected to the DUT)
Interface Level-For an Interface NOT connected to the DUT
Route Ranges Level-List of route ranges for this router
User LSA Groups Level-List of User LSA groups for this router
User LSA Group Level-List of User LSAs for this group

- [OSPFv2 Interfaces Window](#)
- [OSPFv2 Interface Window](#)
- [OSPFv2 Learned LSAs Window](#)

Fig 20.2.1 Shows the contents of OSPFv2 Protocol Tree

5. To Know More on IxNetwork Classic

<https://www.youtube.com/watch?v=gWjgFndvSAI>

<http://openixia.com/sampleScripts//IxNetwork/HighLevelApi/Classic/Tcl>

6. Support

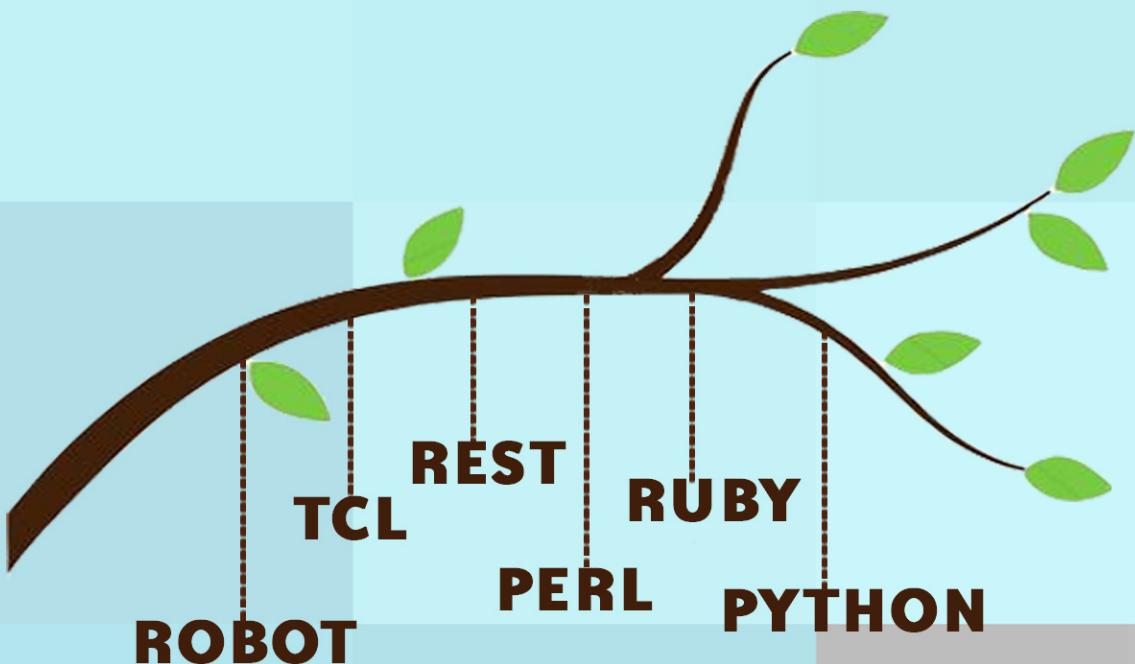
For more information, visit <https://support.ixiacom.com/>

For support assistance, contact support.ix@keysight.com



AUTOMATION

#WeMakeItEasier



<https://github.com/openixia>

For queries : support.ix@keysight.com