**IXNETWORK-CLASSIC**

**QUICK REFERENCE GUIDE**



**PREFACE**

1. **[Overview](#overview1)**
2. **[Configure OSPFv2 through GUI](#configure1)**
   1. [Add Chassis and Reserve Ports](#Add_chasis1)
   2. [Configure Ports](#configure_port1)

* 1. [Emulate OSPFv2 Protocol](#Emulate_ospf1)

* 1. [Configure OSPFv2](#configure_ospf1)

* 1. [Create OSPFv2 Route Ranges](#create_ospf1)

* 1. [Start Protocols](#start_protocol1)
  2. [Check Learned LSAs](#ls1)

* 1. [Configure Traffic](#Configure_traffic1)

* 1. [Add Endpoints to Traffic](#add_endpoints1)

* 1. [Edit Packet](#Edit_packet1)

* 1. [Setup Flow Group](#Setup_flow1)
  2. [Setup Frame Size](#Setup_frame1)
  3. [Setup Line Rate](#Setup_line1)

* 1. [Setup Flow Tracking](#Setup_flow_tracking1)

* 1. [Preview Flow Groups](#preview_flow1)

* 1. [Validate Traffic Items](#validate_traffic1)

* 1. [Apply Traffic, Start Traffic and Statistics View](#apply_traffic1)

1. **[Configure OSPFv2 through Automation](#configure_ospfv2)**

* 1. [Initialize Environment](#initialize)

* 1. [Add Chassis and Reserve Ports](#add_chasis32)

* 1. [Configure Ports](#configure33)

* 1. [Create OSPFv2](#create_ospf134)

* 1. [Create OSPFv2 Route Ranges](#create_ospf35)

* 1. [Start Protocols](#start_protocol36_1_1)
  2. [Check Learned LSAs](#start_protocol36_1_1)

* 1. [Configure Traffic](#Configure_traffic37_1_1)

* 1. [Start and Stop Traffic](#start_stop_traffic_1)

* 1. [Get statistics](#get_stattistics_n)

1. **[Other Utilities](#other_u)**

* 1. [IxNetwork API Documentation Browser](#api_brow)
  2. [Script Gen](#script_gen)

* 1. [F1 Option](#f1_optio)

1. **[To Know More on IxNetwork-Classic](#know_more)**
2. **[Support](#support)**

1. **[Overview:](#overview)**

* Ix-Network Classic is specifically targeted for the performance and functionality testing of high-speed, high-capacity routers and 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.

1. **[Configure OSPFv2 through GUI:](#configure)**

This section visualizes the scenario to configure OSPFv2 emulation, verify end-end traffic flow and its statistics.

* 1. **[Add Chassis and Reserve Ports](#Add_chassis)**

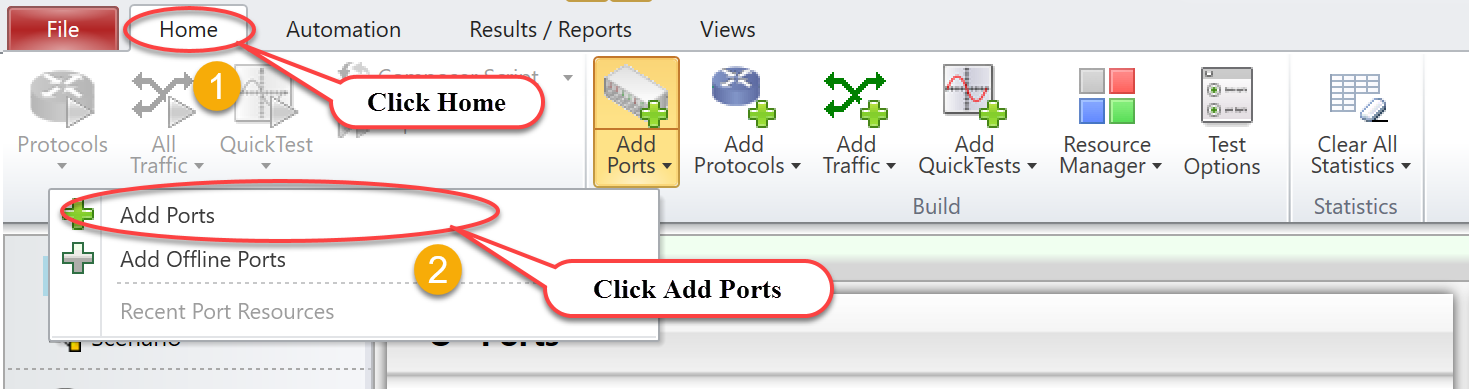


Fig 1.1 Add Ports

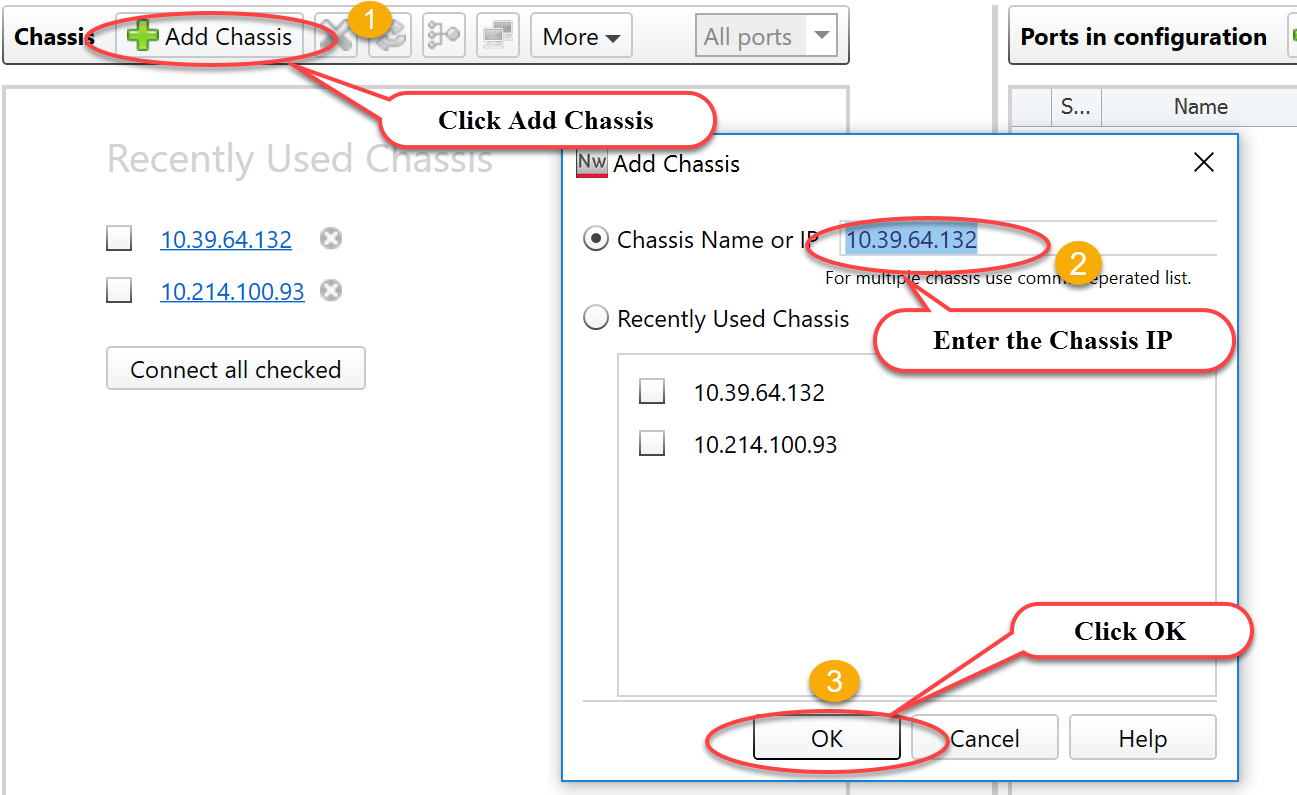


Fig 1.2: Add Chassis to reserve ports

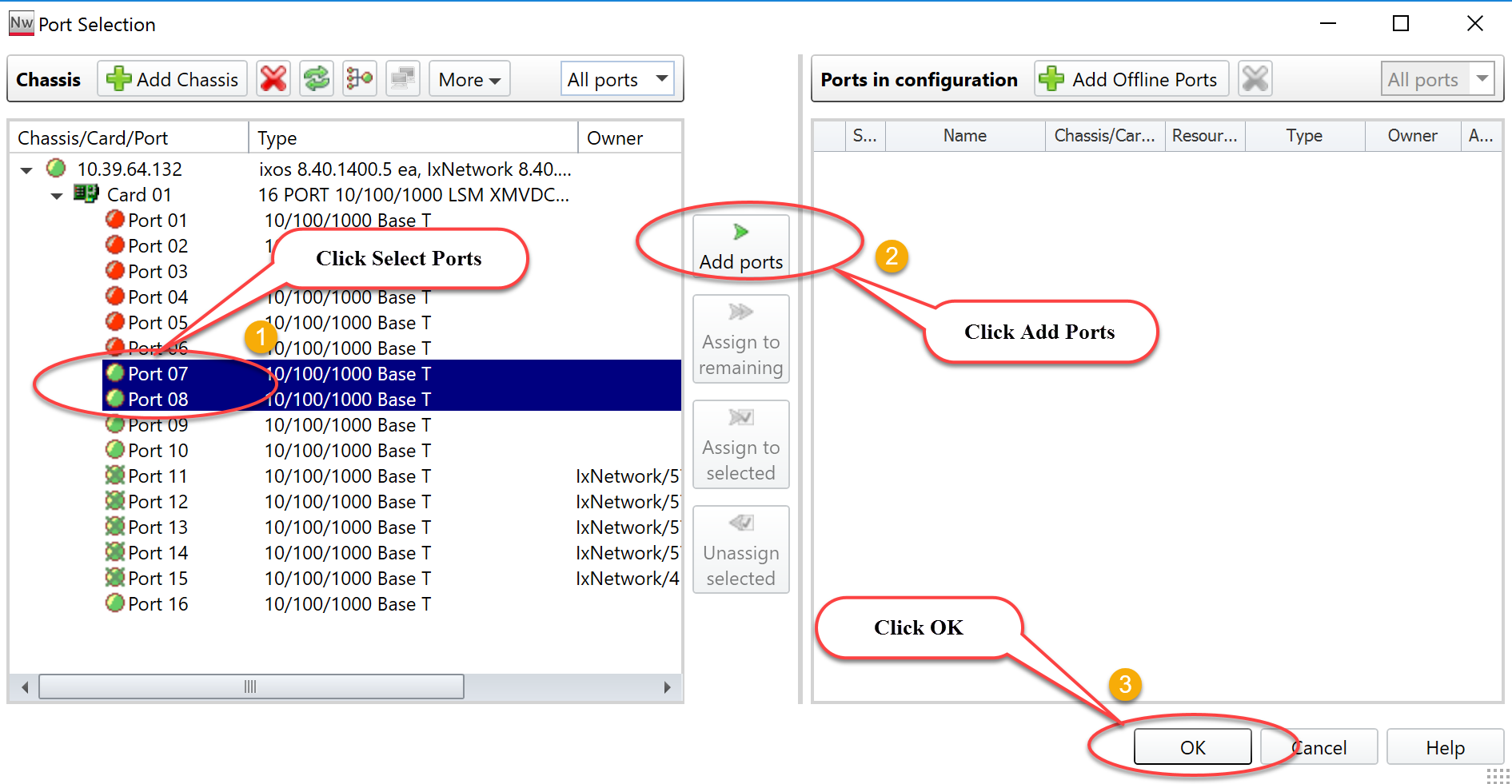


Fig 1.3: Select Ports and Reserve Ports

* 1. **[Configure Ports](#configure_port)**

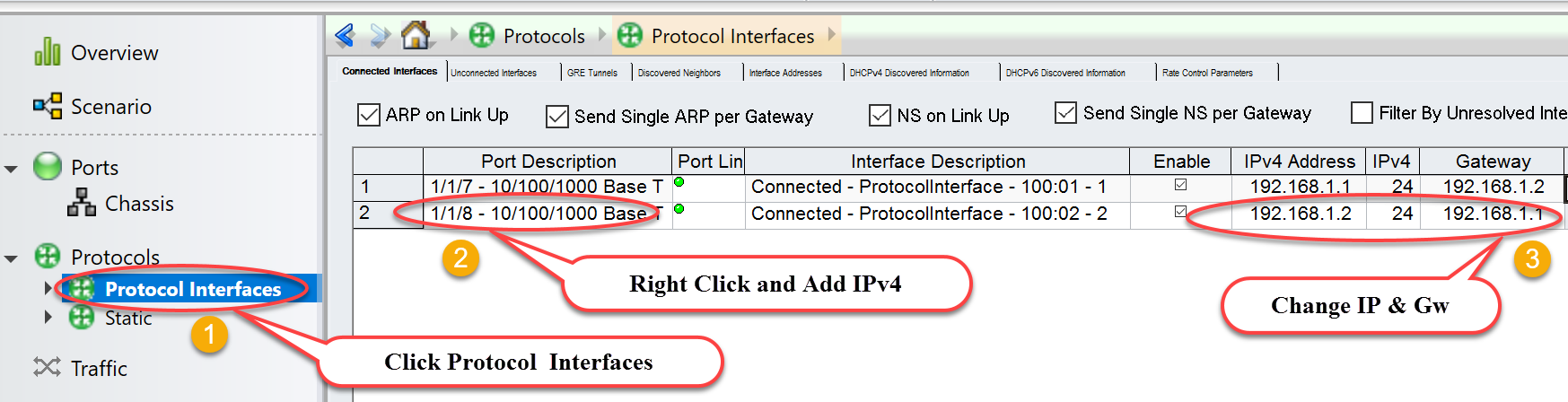


Fig 2.1: Configure Port’s attributes

* 1. **[Emulate OSPFv2 Protocol](#Emulate_ospf)**

Enable OSPFv2 on interfaces.

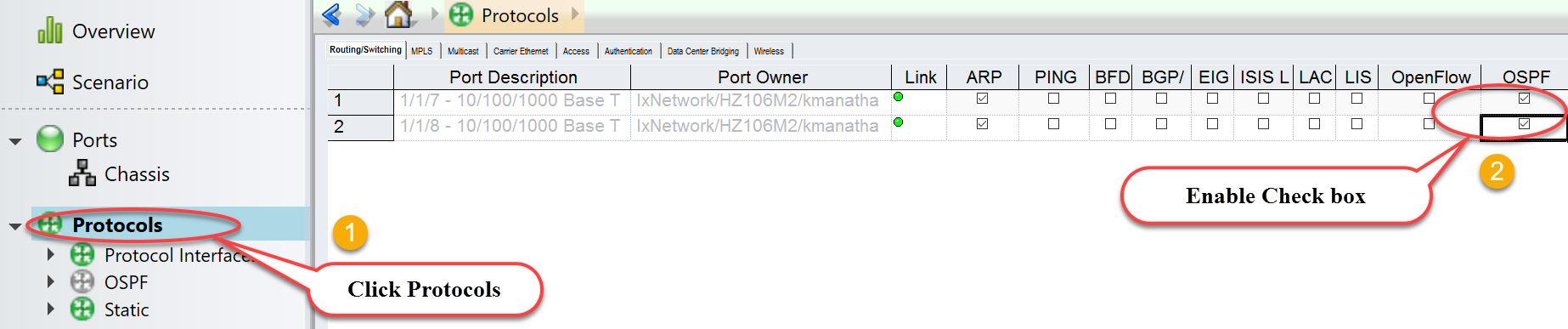


Fig 3.1: Emulate OSPFv2

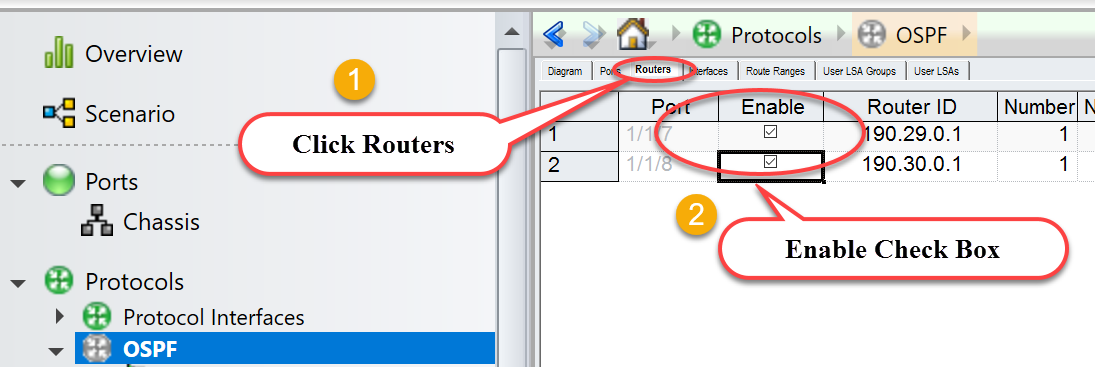


Fig 3.2: Enable OSPFv2 on interfaces

* 1. **[Configure OSPFv2](#configure_ospf)**

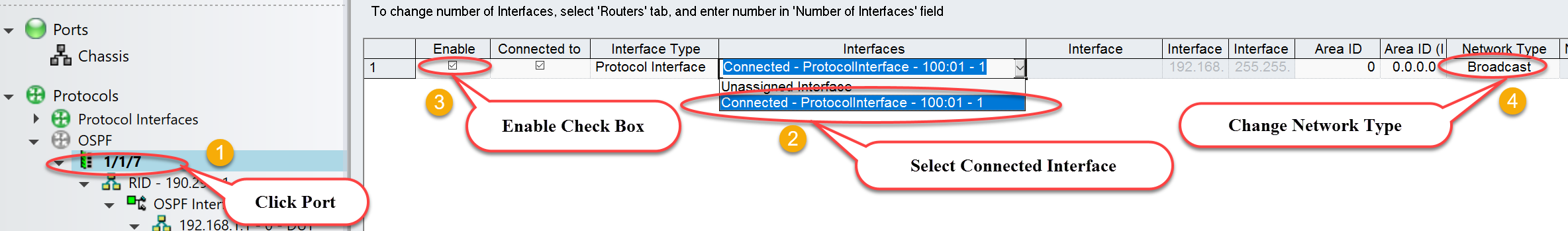


Fig 4.1: Map connected interfaces to configure OSPFv2 attributes

* 1. **[Create OSPFv2 Route Ranges](#create_ospf)**

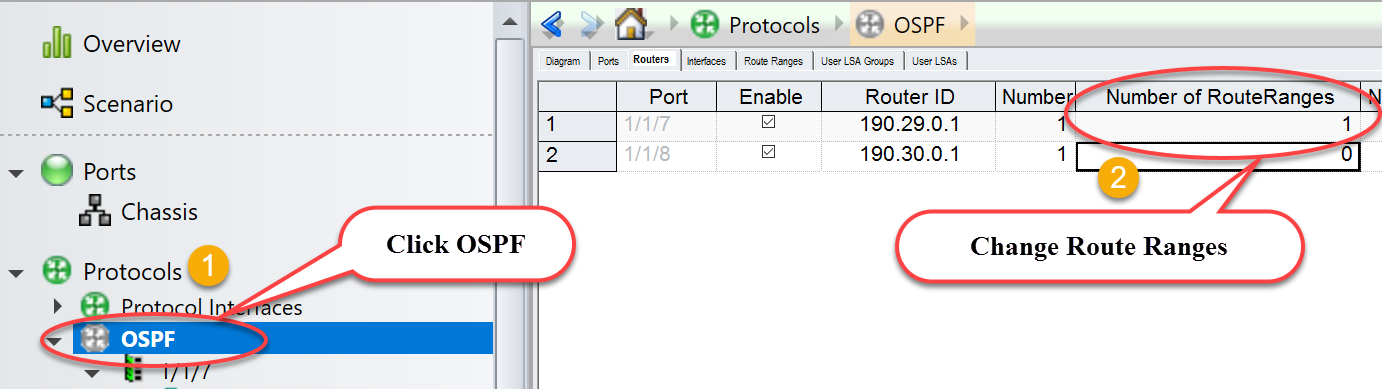


Fig 5.1: Configure number of route ranges required

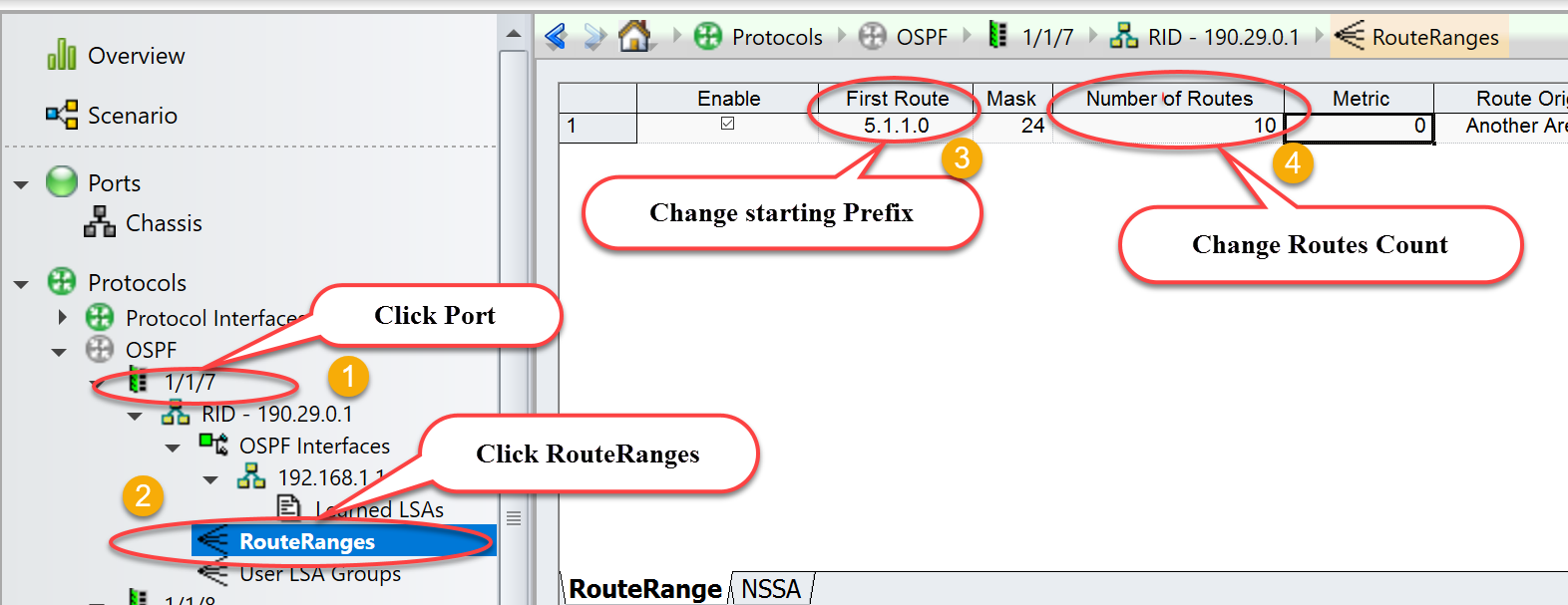


Fig 5.2: Create Ipv4 prefix pool on selected port 1/1/7

* 1. **[Start Protocols](#start_protocol)**

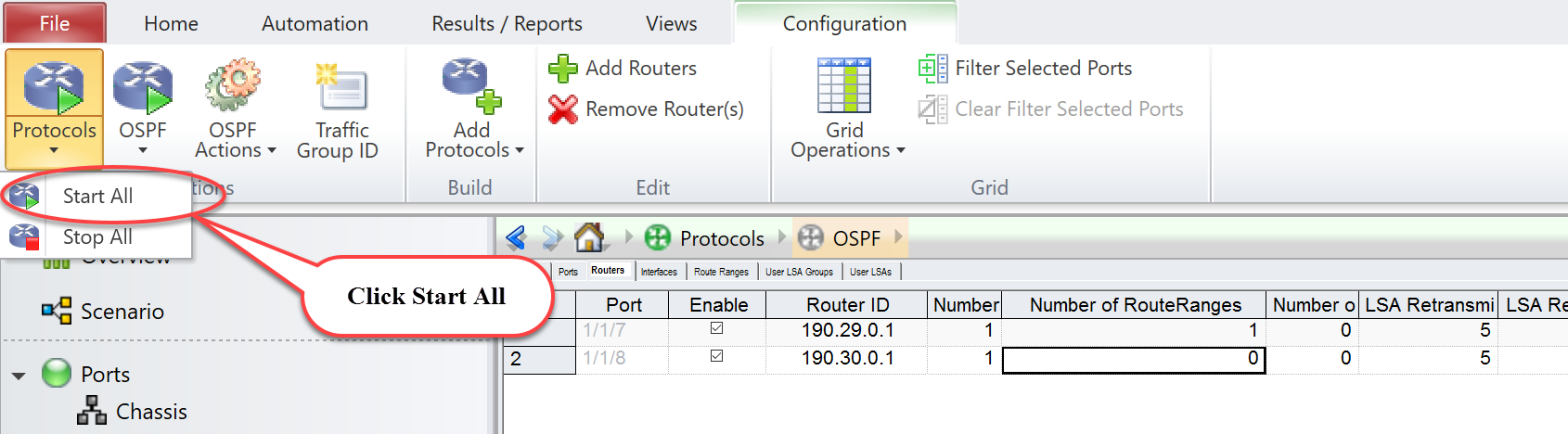


Fig 6.1: Start all protocols

* 1. [Check Learned LSAs](#TOC)

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

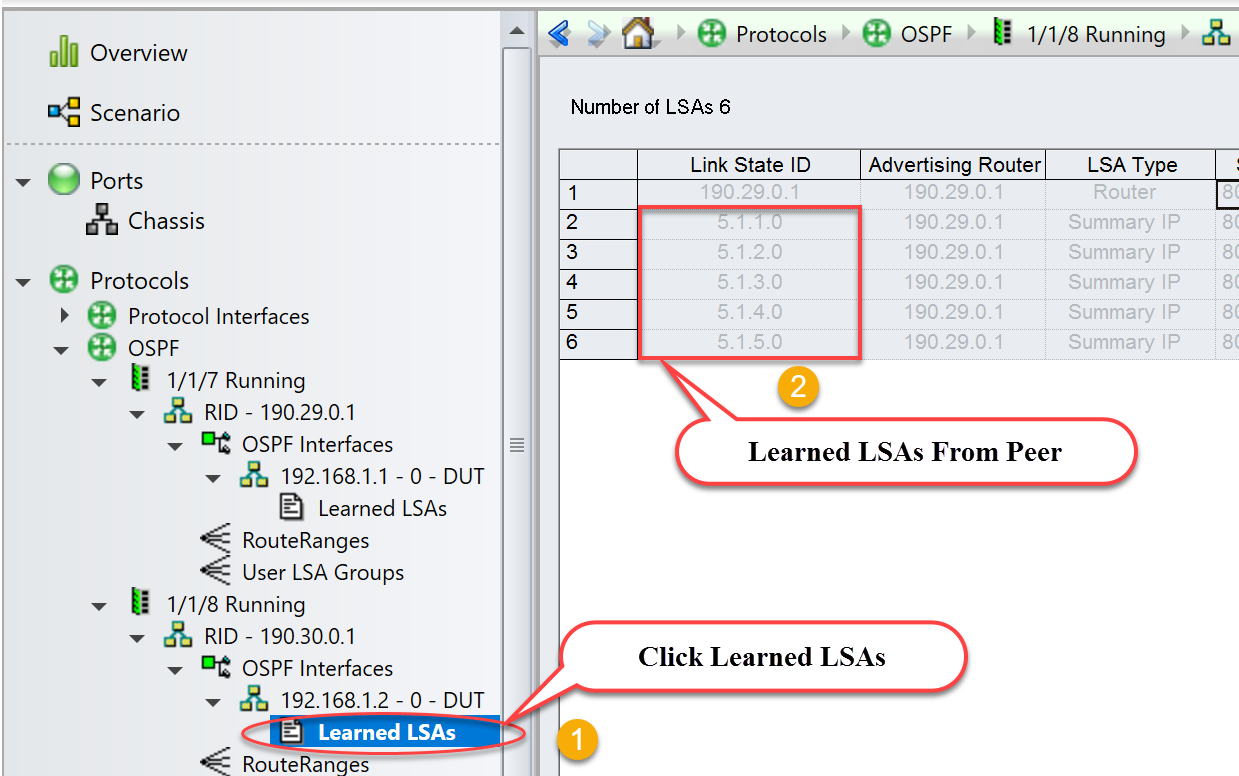
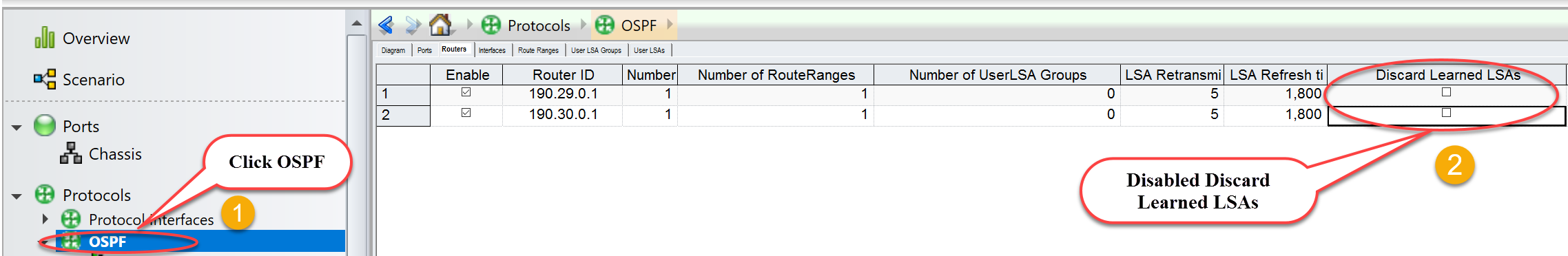


Fig 7.1 Disable discard Learned LSA and check Learned LSAs on 1/1/8 from peer router 1/1/7

* 1. **[Configure Traffic](#Configure_traffic)**

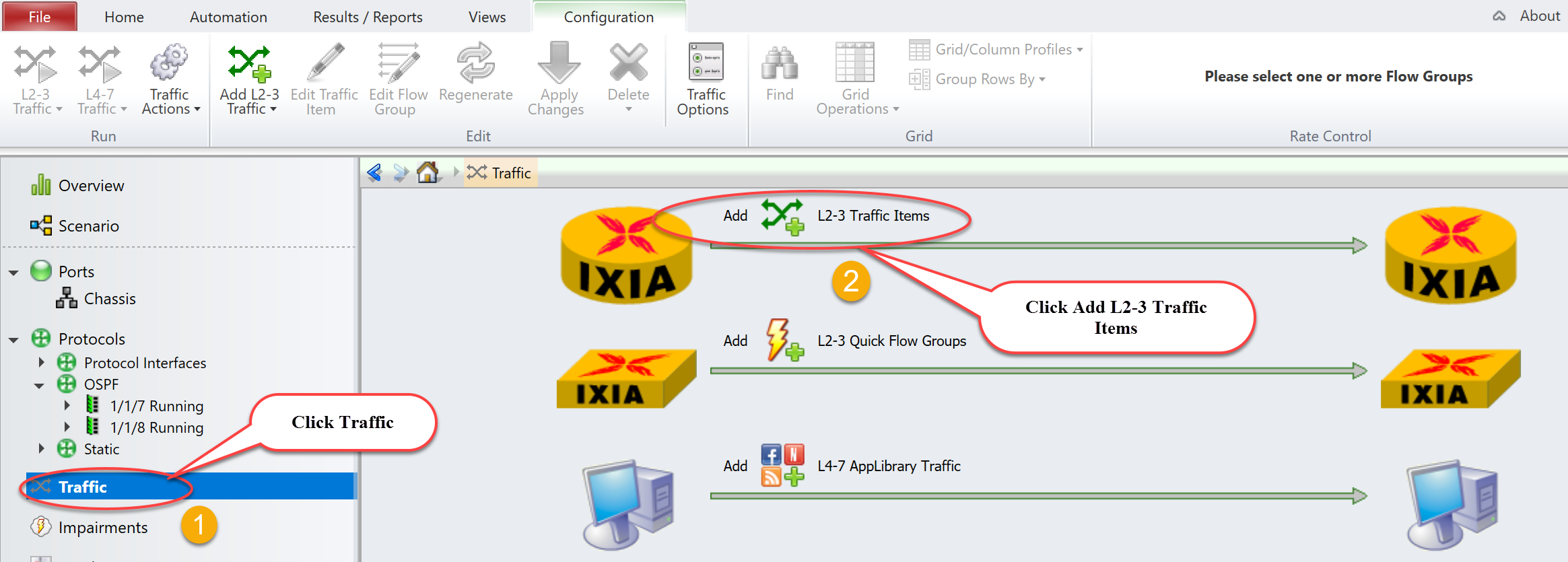


Fig 8.1: Create traffic stream

* 1. **[Add Endpoints To Traffic](#add_endpoints)**

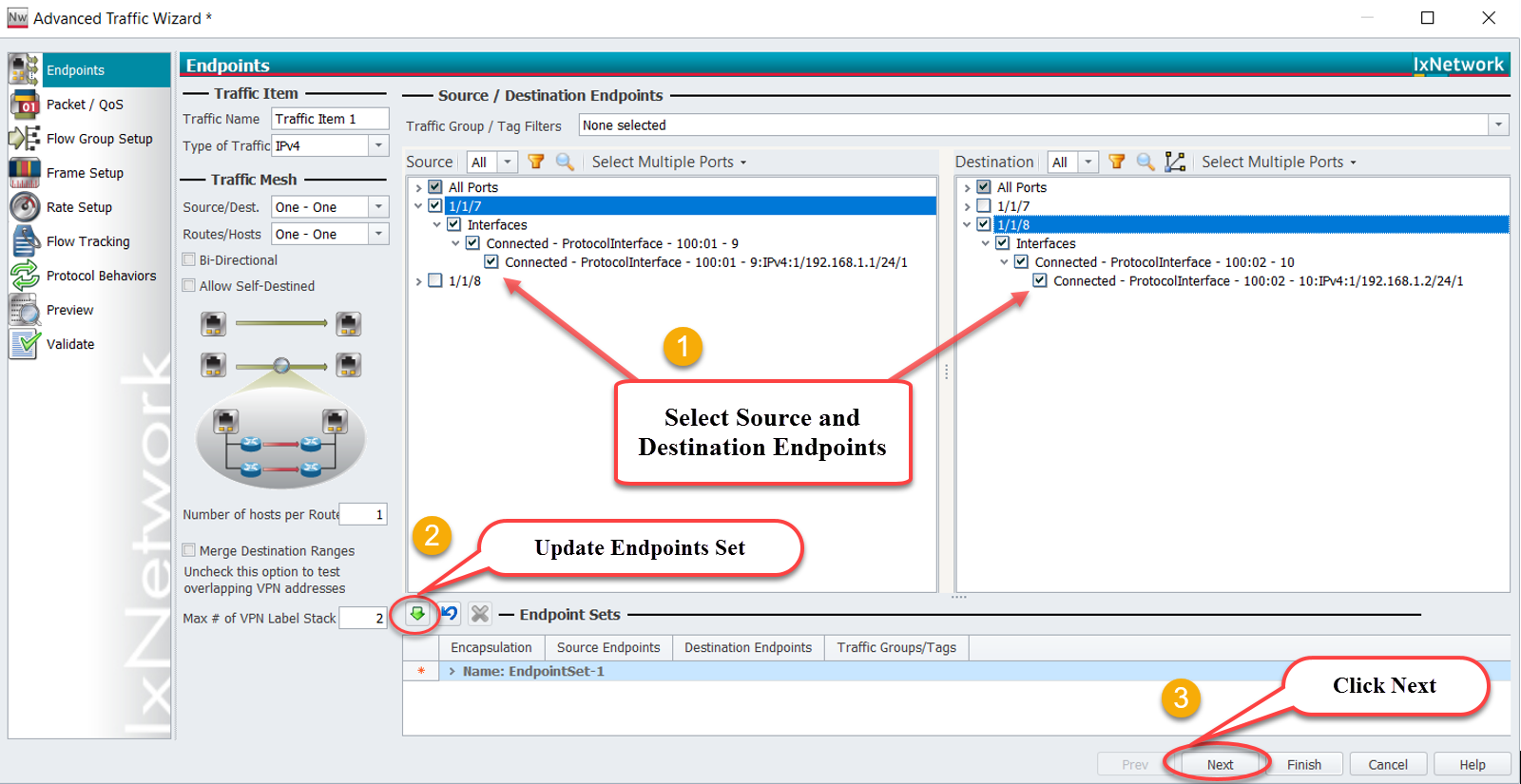


Fig 8.2: Setup source and destination endpoints

* 1. **[Edit Packet](#Edit_packet)**

\*Edit Packet and Flow Group Setup are optional.

\*Optional step. Use the Advanced Traffic Wizard-Packet/QoS dialog to view endpoint sets, configure packet fields and Quality of Service (QoS) fields.

Fig 8.3: Edit packet header

* 1. [**Setup Flow Group**](#Setup_flow)

\*Optional step. Use window to create flows groups that help to create groups of flows based on selectable packet fields. One flow group/high-level stream is created for each selected field.

Fig 8.4: Setup flow group

* 1. **[Setup Frame Size](#Setup_frame)**

\*Setup frame Size and Line rate are optional.

Fig 8.5: Setup Frame size as per test scenario

* 1. **[Setup Line Rate](#Setup_line)**

8.6: Setup line rate

* 1. **[Setup Flow Tracking](#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

* 1. [**Preview Flow Groups**](#preview_flow)

Fig 8.9 View flow group which is currently configured

* 1. [**Validate Traffic Items**](#validate_traffic)

Fig 8.10 Validate the traffic item to identify errors

* 1. **Apply Traffic, Start Traffic and Statistics View:**

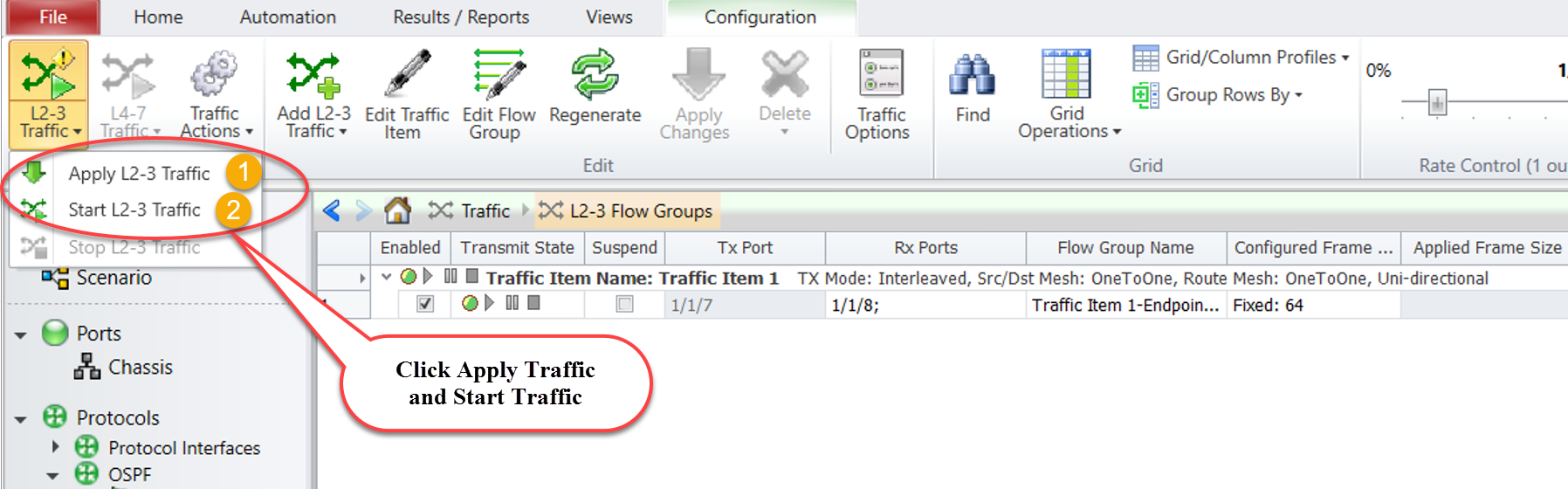


Fig 9.1 Apply and start traffic

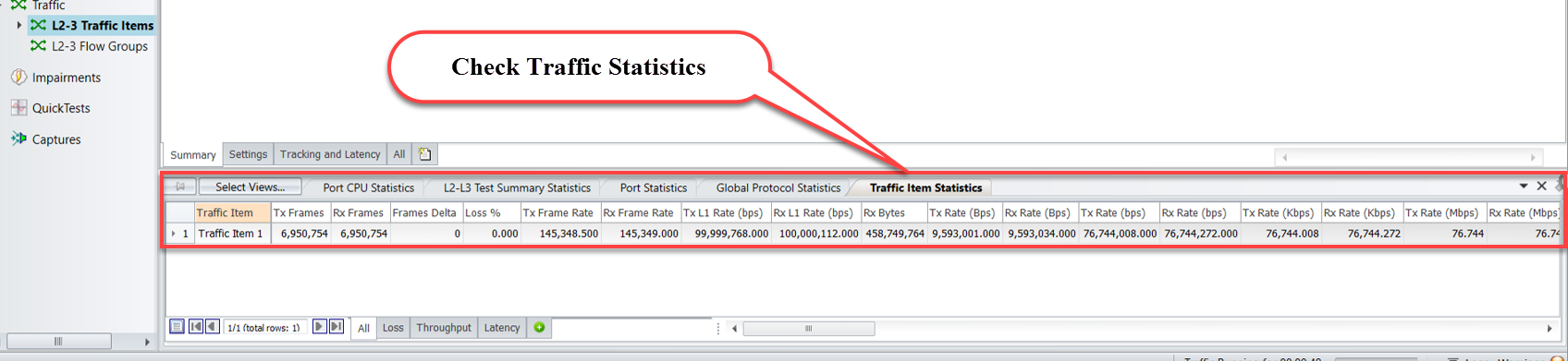


Fig 9.2 Verify traffic statistics

1. **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.

* 1. **[Initialize Environment](#initialize1)**

Source Ixia package and proceed with HLTAPI execution.

|  |  |
| --- | --- |
| ***TCL***  ***> 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** | ***PYTHON***  ***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\_autopath=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\_autopath=tcl\_dependencies)***  ***ixiatcl = IxiaTcl(); ixia = IxiaHlt(ixiaTcl)*** |

* 1. **[Add Chassis and Reserve Ports](#add_chasis32_1)**

**::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.

|  |  |
| --- | --- |
| ***TCL***  ***> set connect\_result [::ixia::connect \***  ***-device 10.39.64.132 \***  ***-port\_list {1/7 1/8} \***  ***-username "ixia\_classic" \***  ***-reset 1 \***  ***-ixnetwork\_tcl\_server 10.154.161.223:8009 \***  ***-tcl\_server 10.154.161.223]***  ***> set ports [keylget connect\_result vport\_list]***  ***> set port1 [lindex $ports 0]***  ***> set port2 [lindex $ports 1]*** | ***PYTHON***  ***> connect\_result = ixia.connect ( \***  ***ixnetwork\_tcl\_server = "10.154.161.223:8009", \***  ***tcl\_server = 10.154.161.223, \***  ***device = 10.39.64.132, \***  ***port\_list = "1/7 1/8", \***  ***break\_locks= 1, \***  ***reset = 1)***  ***> ports = connect\_result['vport\_list'].split()***  ***> port1 = ports[0]***  ***> port2 = ports[1]*** |

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

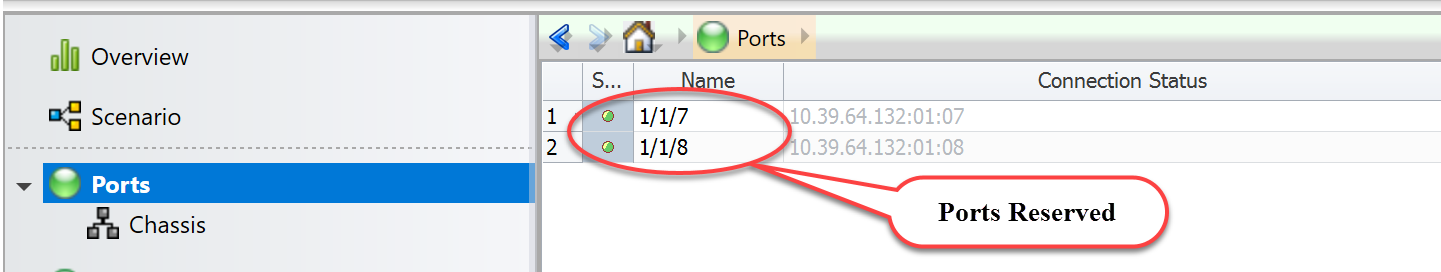


Fig 10.1 Connected to chassis and reserved ports

* 1. **[Configure Ports](#configure33_1)**

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

|  |  |
| --- | --- |
| ***TCL***  ***> 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]***  ***> 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]*** | ***PYTHON***  ***> 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')***  ***> 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')*** |

Fig 11.1 Configure ports attributes using HLAPI

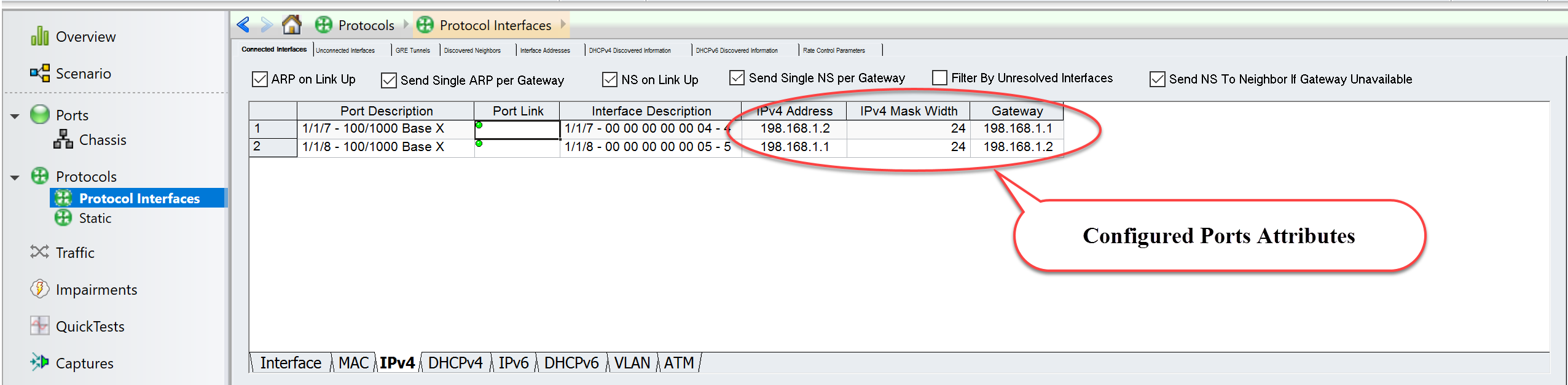


Fig 11.2 View configured attributes in GUI

* 1. **[Create OSPFv2](#configure_ospfv2_34)**

**::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.

|  |  |
| --- | --- |
| ***TCL***  ***> set ospf\_emul\_res1 [::ixia::emulation\_ospf\_config \***  ***-port\_handle $port1 -mode create \***  ***-session\_type ospfv2 \ -area\_id 0.0.0.1 -router\_id 1.1.1.1 \***  ***-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 [keylget ospf\_emul\_res1 handle]***  ***> set ospf\_emul\_res2 [::ixia::emulation\_ospf\_config \***  ***-port\_handle $port2 -mode create \***  ***-session\_type ospfv2 \***  ***-area\_id 0.0.0.1 -router\_id 2.2.2.2 \***  ***-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 [keylget ospf\_emul\_res2 handle]*** | ***PYTHON***  ***> ospf\_emul\_res1 = ixia.emulation\_ospf\_config ( \***  ***port\_handle = port1, mode = create, \***  ***session\_type = 'ospfv2', \ area\_id = 0.0.0.1, router\_id = 1.1.1.1, \***  ***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']***  ***> ospf\_emul\_res2 = ixia.emulation\_ospf\_config ( \***  ***port\_handle = port2, mode = create, \***  ***session\_type = 'ospfv2', \***  ***area\_id = 0.0.0.1, router\_id = 2.2.2.2, \***  ***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']*** |

Fig 12.1 Configure OSPFv2 attributes using HLAPI

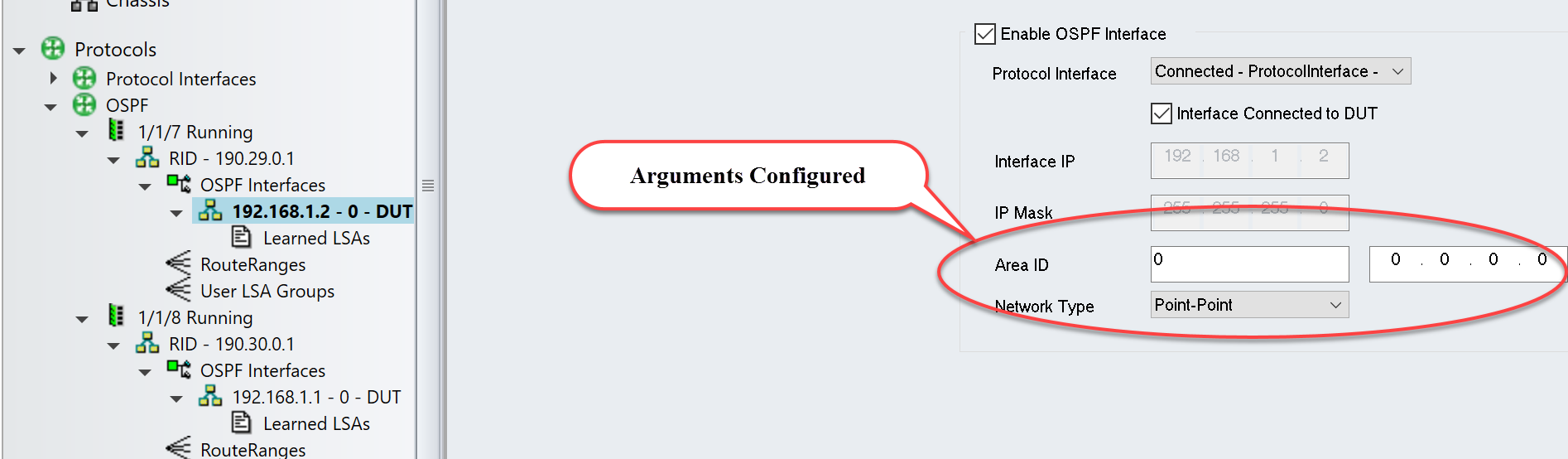


Fig 12.2 View configured ospfv2 attributes on 1/1/7 in GUI

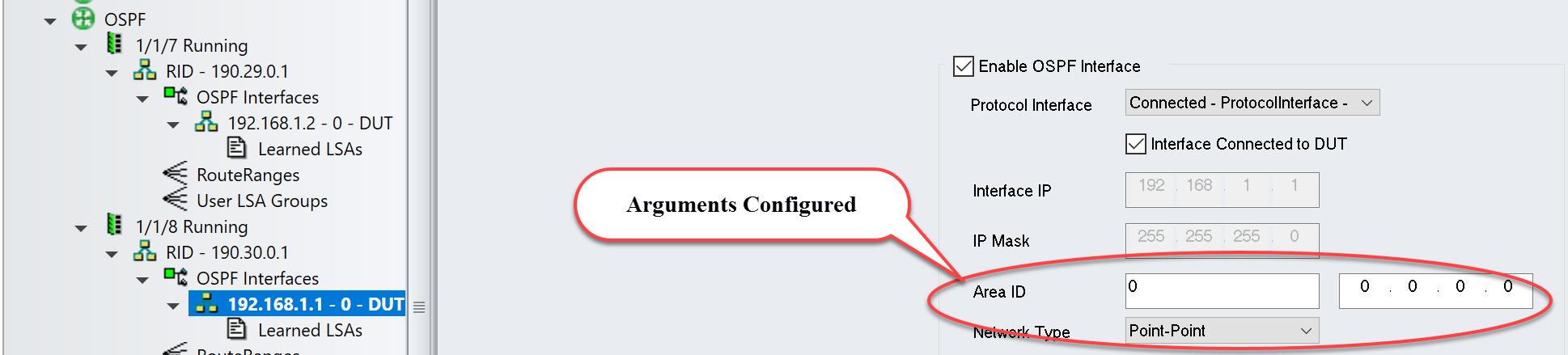


Fig 12.3 View configured ospfv3 attributes on 1/1/8 in GUI

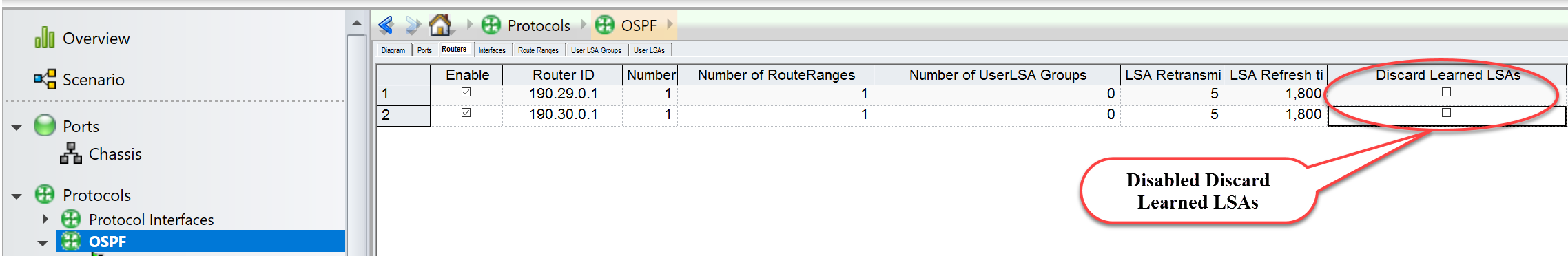


Fig 12.4 View Discard Learned LSAs disbaled in GUI

* 1. **[Create OSPFv2 Route Ranges](#create_ospf35_1)**

**::ixia::emulation\_ospf\_topology\_route\_config** - Add OSPFv2 route(s) to a simulated OSPFv2 router interface.

|  |  |
| --- | --- |
| ***TCL***  ***> 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]*** | ***PYTHON***  ***> 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)*** |

Fig 13.1 Configure OSPFv2 Route Ranges using HLAPI

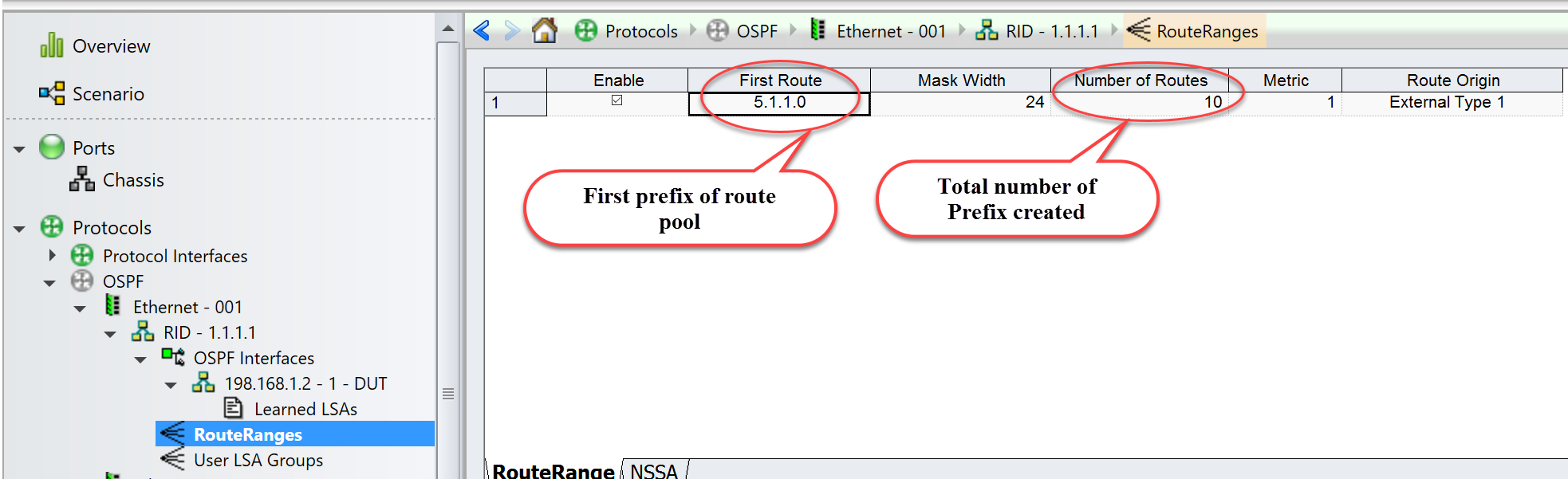


Fig 13.2 View configured route ranges in GUI

* 1. **[Start Protocols:](#start_protocol36_1)**

**::ixia::emulation\_ospf\_control**  - Start OSPF protocol.

|  |  |
| --- | --- |
| ***TCL***  ***> set result1 [::ixia::emulation\_ospf\_control \***  ***-handle $ospf\_handle1 \***  ***-port\_handle $port1 \***  ***-mode start]***  ***> set result2 [::ixia::emulation\_ospf\_control \***  ***-handle $ospf\_handle2 \***  ***-port\_handle $port2 \***  ***-mode start]***  ***> puts "Waiting for sometime for the protocols to converge"***  ***> sleep 45*** | ***PYTHON***  ***> result1 = ixia.emulation\_ospf\_control ( \***  ***handle = ospf\_handle1, \***  ***port\_handle = port1, \***  ***mode = 'start')***  ***> result2 = ixia.emulation\_ospf\_control ( \***  ***handle = ospf\_handle2, \***  ***port\_handle = port2, \***  ***mode = 'start')***  ***> print """Waiting for sometime for the protocols to converge"""***  ***> time.sleep(45)*** |

Fig 14.1 Start OSPFv2 protocols using HLAPI

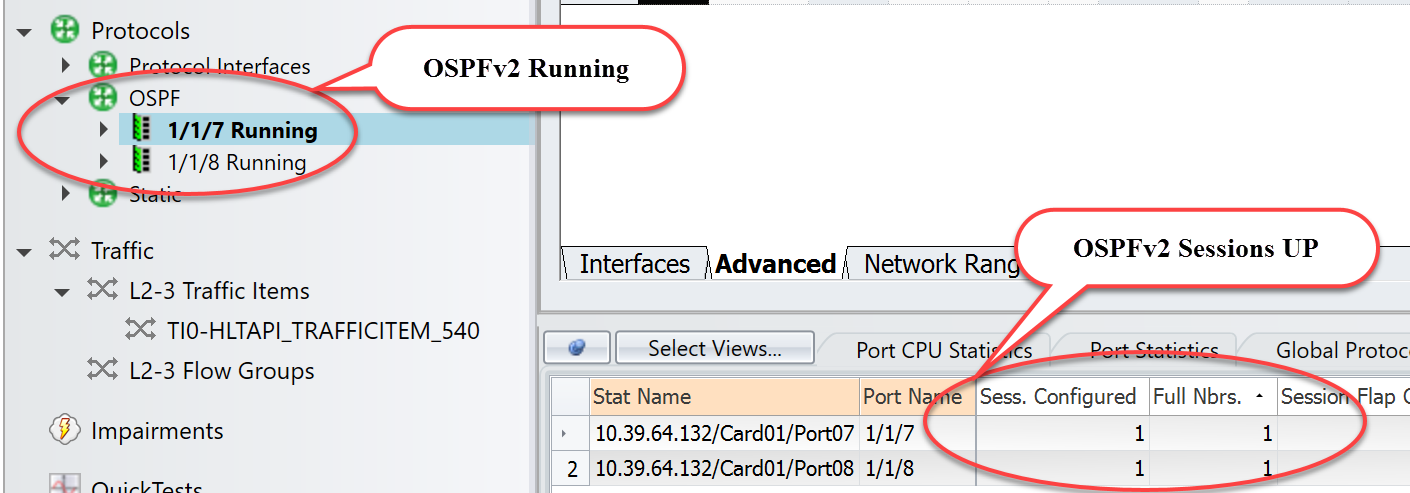


Fig 14.2 View ospfv2 session states details in GUI

* 1. **[Check Learned LSAs:](#start_protocol36_1)**

**::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]*** | ***PYTHON***  ***> learned\_info = ixia.emulation\_ospf\_info ( \***  ***mode = 'learned\_info', \***  ***handle = ospf\_handle2)*** |

Fig 15.1 Retrieve learned LSA info using HLAPI

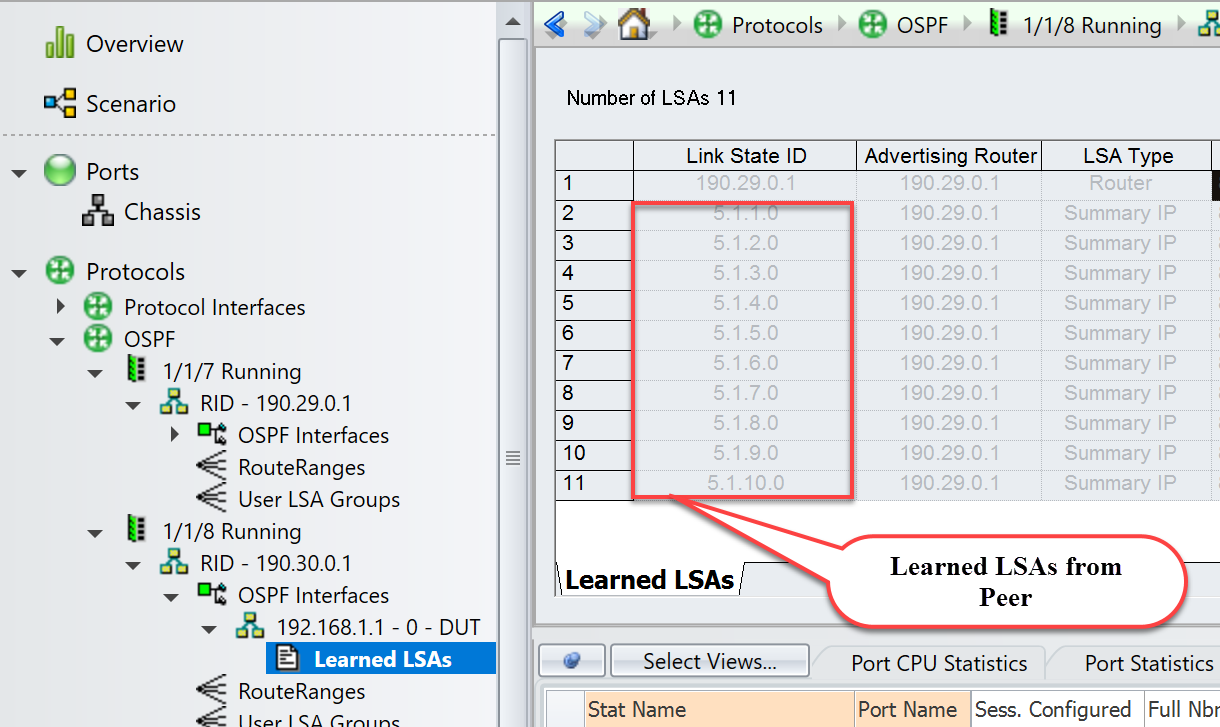


Fig 15.2: View learned LSAs info in GUI

* 1. **[Configure Traffic:](#Configure_traffic37_1)**

**::ixia::traffic\_config** - Configures traffic streams on the specified ports with the specified options.

|  |  |
| --- | --- |
| ***TCL***  ***> 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]*** | ***PYTHON***  ***> traffic\_res = ixia.traffic\_config ( \***  ***mode = 'create', \***  ***transmit\_mode = 'continuos', \***  ***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)*** |

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

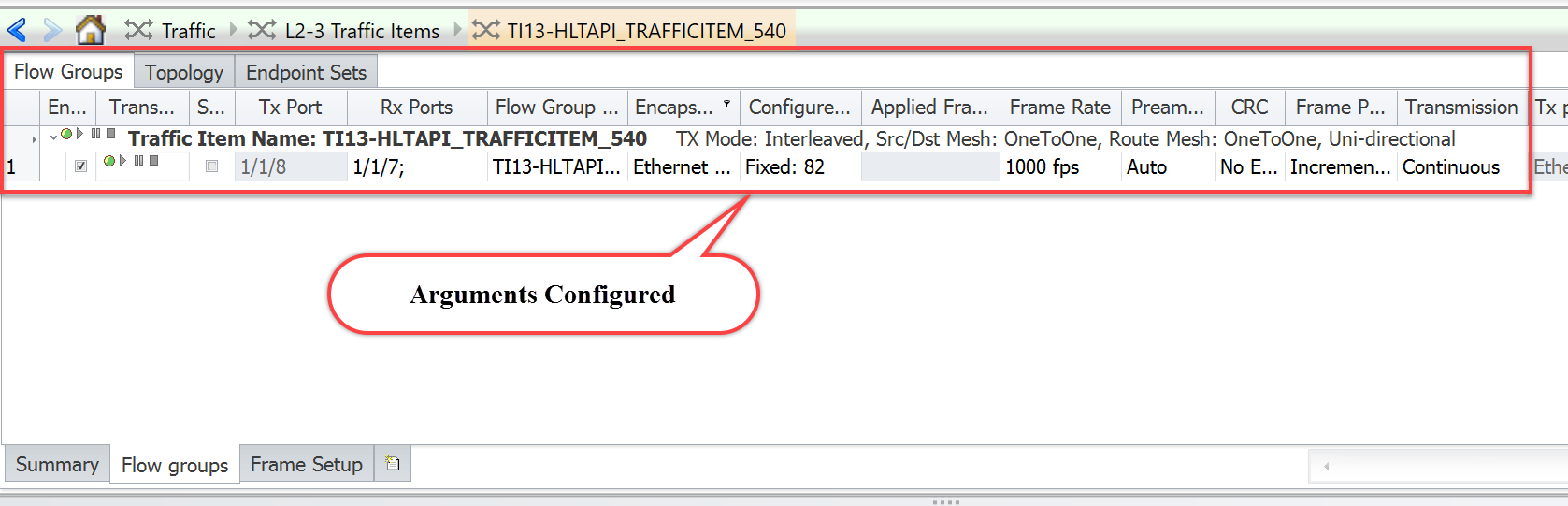


Fig 16.2: View traffic item configuration in GUI

* 1. **[Start and Stop Traffic:](#start_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.

|  |  |
| --- | --- |
| ***TCL***  ***> ::ixia::traffic\_control -action run***  ***> puts "Waiting for sometime for the traffic to flow"***  ***> sleep 30***  ***> ::ixia::traffic\_control -action stop*** | ***PYTHON***  ***> ixia.traffic\_control (action = 'run')***  ***> print "Waiting for sometime for the traffic to flow"***  ***> time.sleep(30)***  ***> ixia.traffic\_control (action = 'stop')*** |

Fig 17.1: Start and stop the traffic using HLAPI

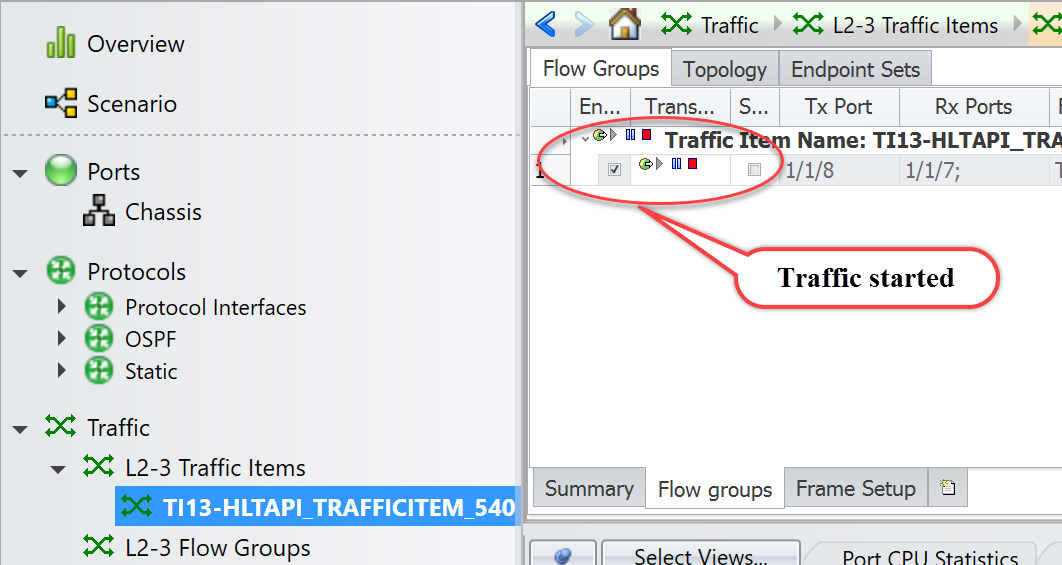


Fig 17.2: View traffic start status in GUI

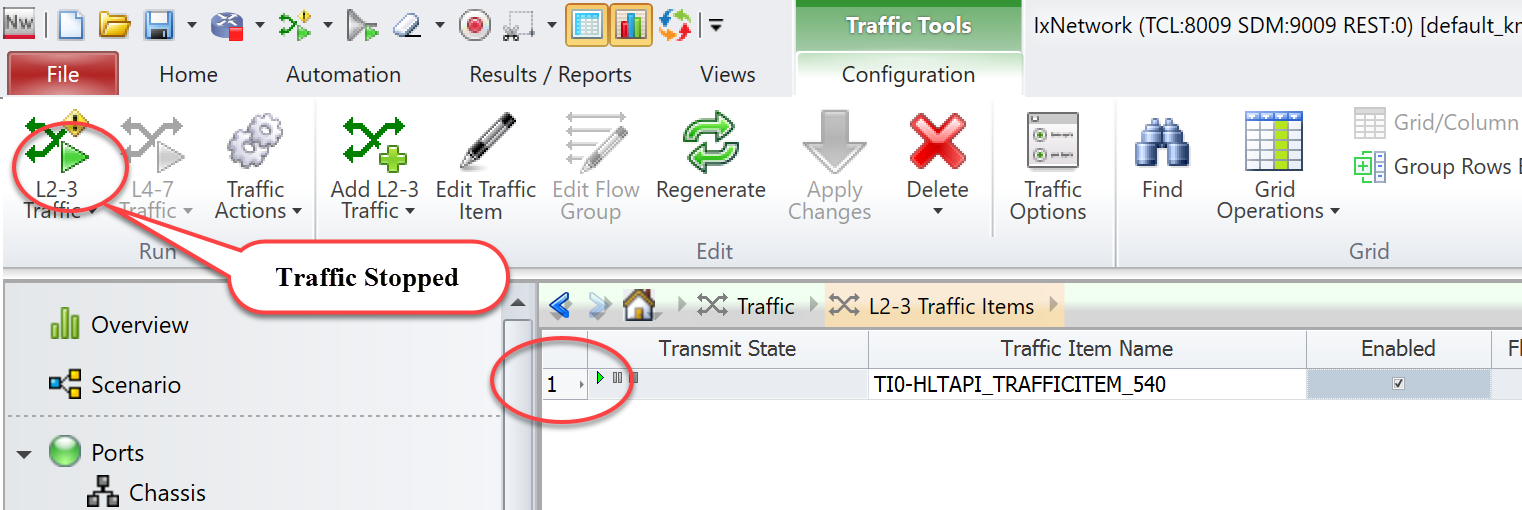


Fig 17.3: View traffic stop status in GUI

* 1. **[Get Statistics:](#get_statistic)**

**::ixia::traffic\_stats -** Gathers statistics depends on the mode such as per\_port\_flows, session, stream, streams, all etc., Below code used mods as ‘all’.

|  |  |
| --- | --- |
| ***TCL***  ***> set stats [::ixia::traffic\_stats \***  ***-mode all \***  ***-traffic\_generator "ixnetwork\_540" \***  ***-measure\_mode mixed]*** | ***PYTHON***  ***> stats = ixia.traffic\_stats ( \***  ***mode = all, \***  ***traffic\_generator = "ixnetwork\_540" \***  ***measure\_mode = mixed)*** |

Fig 18.1: Retrieve traffic stats using HLAPI

Fig 18.2: Graphical view of traffic statistics

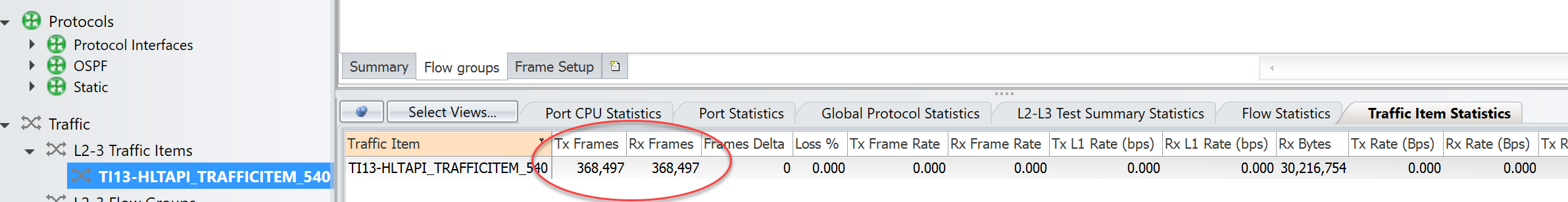


Fig 18.3: Verify Tx Frames and Rx Frames in GUI

1. **[Other Utilities:](#other_u_1)**

**4.1** **[IxNetwork API Documentation Browser:](#api_brow_1)**

* 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 on clicking on BROWSE.

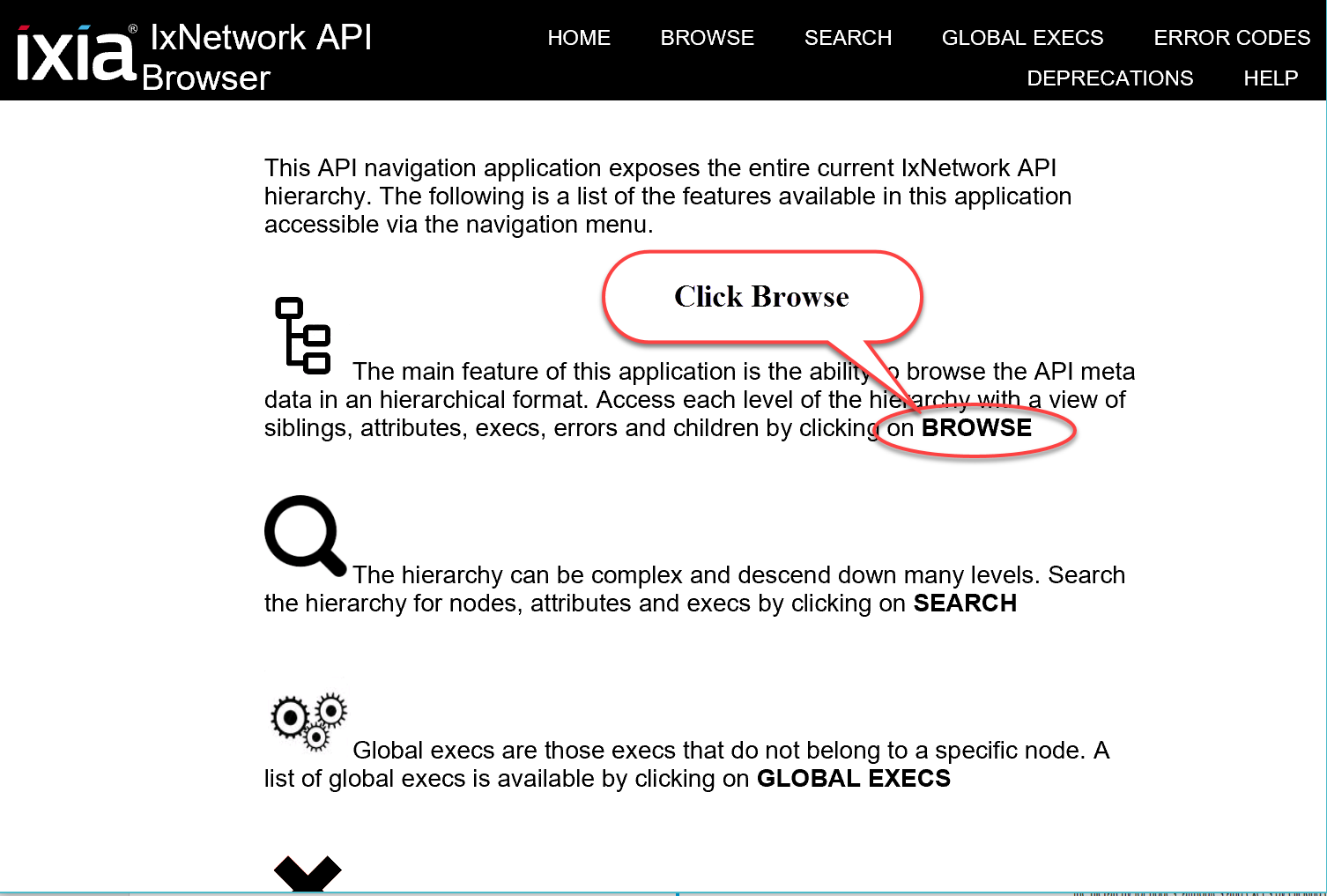


Fig 19.1: IxNetwork API Documentation Browser

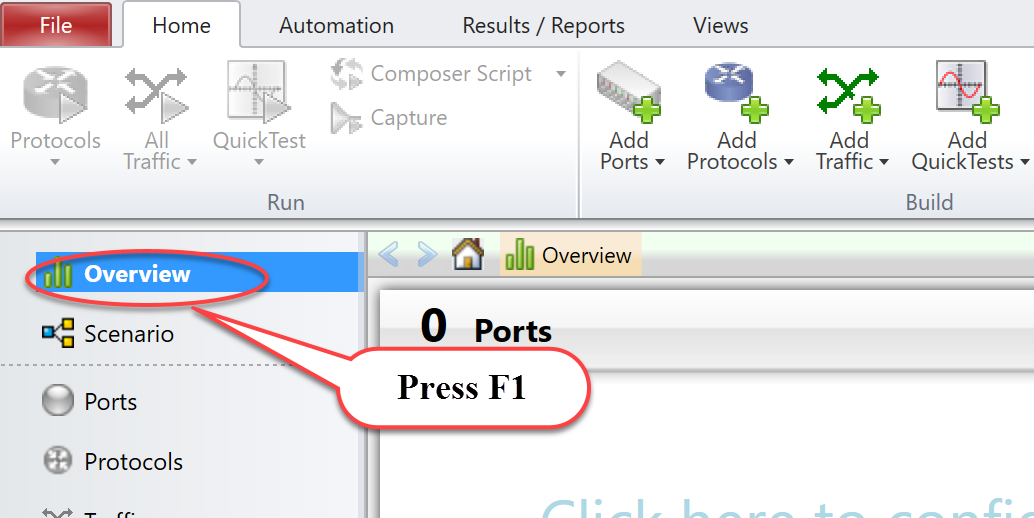
**4.2** **[Script Gen:](#script_gen_1)**

* 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:**](#f1_optio_12)

* 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.



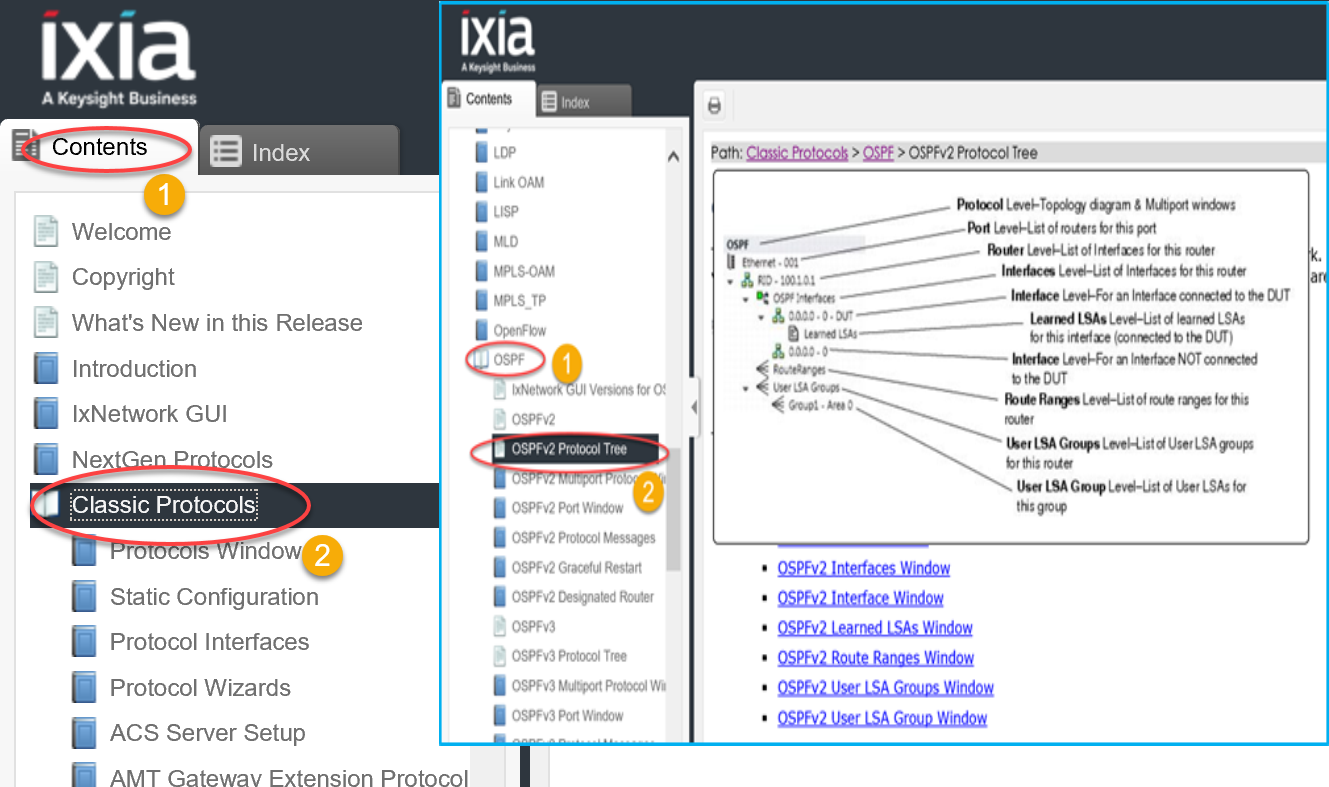


Fig 21.1: Help page to explain about each field in GUI

1. **[To Know More on IxNetwork Classic:](#know_more_)**

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

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

1. **Support**

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

For support assistance, contact [support-india@ixiacom.com](mailto:support-india@ixiacom.com)

