IXNETWORK-NGPF

QUICK REFERENCE GUIDE

TABLE OF CONTENTS

1. Overview:

- NextGen Protocol Framework (NGPF) is Ixia’s new protocol framework.
- Upgraded from the classic protocol framework.
- Built to meet and stay ahead of customer requirements in flexibility and scalability.
- Designed to provide consistent and visual workflow across all protocols.
- Designed to more closely simulate dynamic customer environments.
- Industry leading access,routing and SDN protocol coverage.
- Realistic subscriber emulation of mixed single and dual-stack subscribers.
- Flexibility of scaling the number of emulated devices by using the multiplier feature.
- Granular session control by using configuration grids.
- System level statistics dashboards with on-demand drill-downs.
- Comparable feature set with IxN2X.

2. Configure BGP through GUI:

This section will walk through a scenario which configures BGP emulation manually to get the user introduced with most of the basic features of NGPF.

2.1 Add Chassis and Lock Ports:

- The Port Selection window allows you to manage the ports

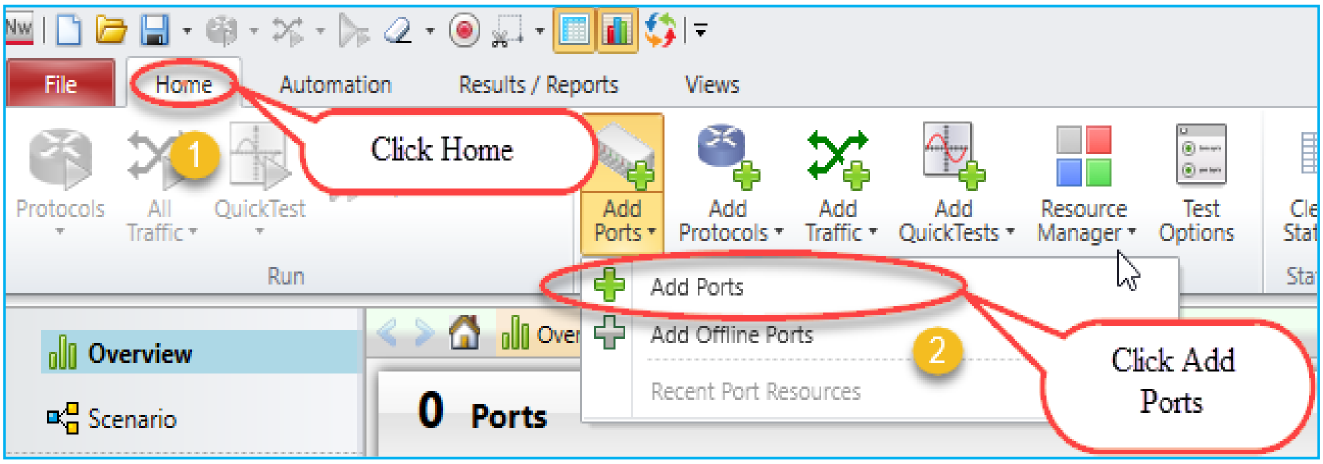
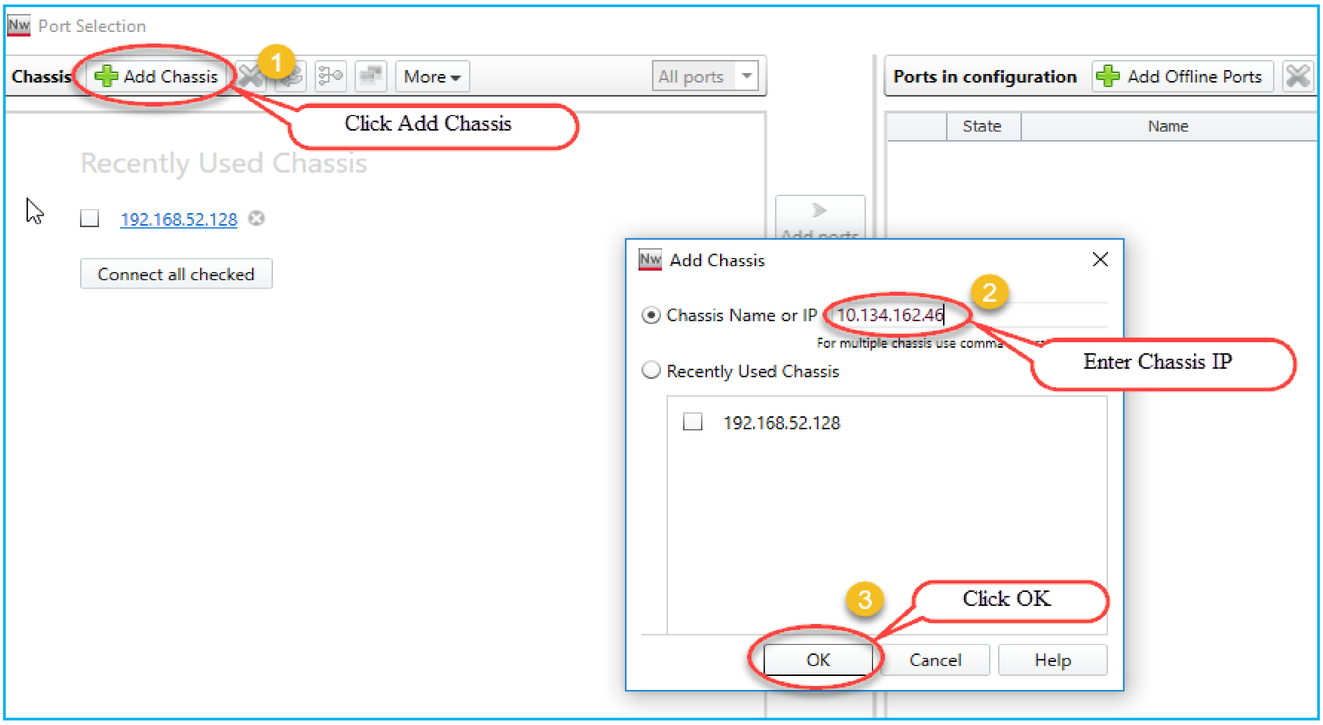


Fig 2.1 Selecting Ports

- Select chassis by entering chassis IP or select chassis from the list of recently used chassis and click **Connect all checked** to add them to the configuration.



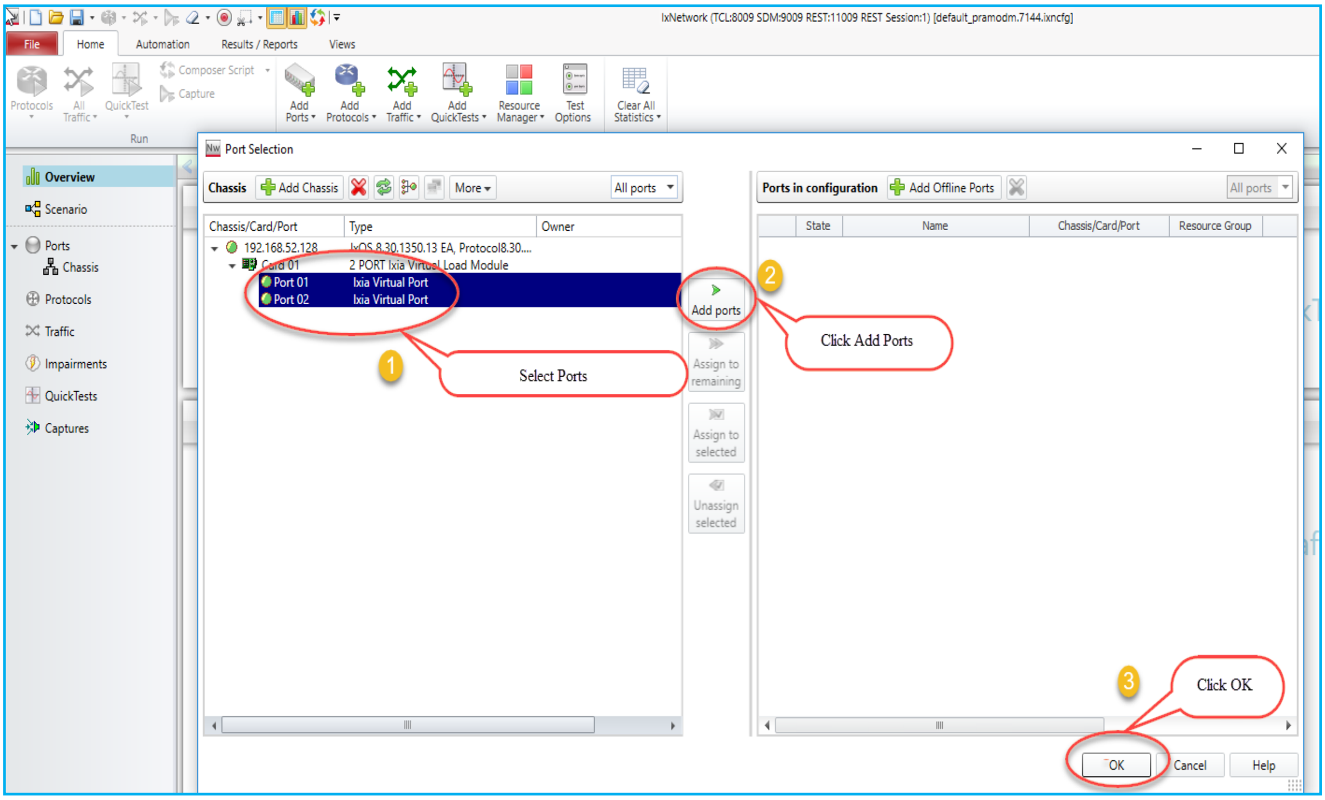
○ Chassis Name or IP 10.134.162.46

1 Click Add Chassis

2 Enter Chassis IP

3 Click OK

Fig 2.1.1 Selects chassis by entering chassis IP



Chassis/Card/Port	Type	Owner
192.168.52.128	IxOS 8.30.1350.13 EA, Protocol8.30...	
Card 01	2 PORT Ixia Virtual Load Module	
Port 01	Ixia Virtual Port	
Port 02	Ixia Virtual Port	

1 Select Ports

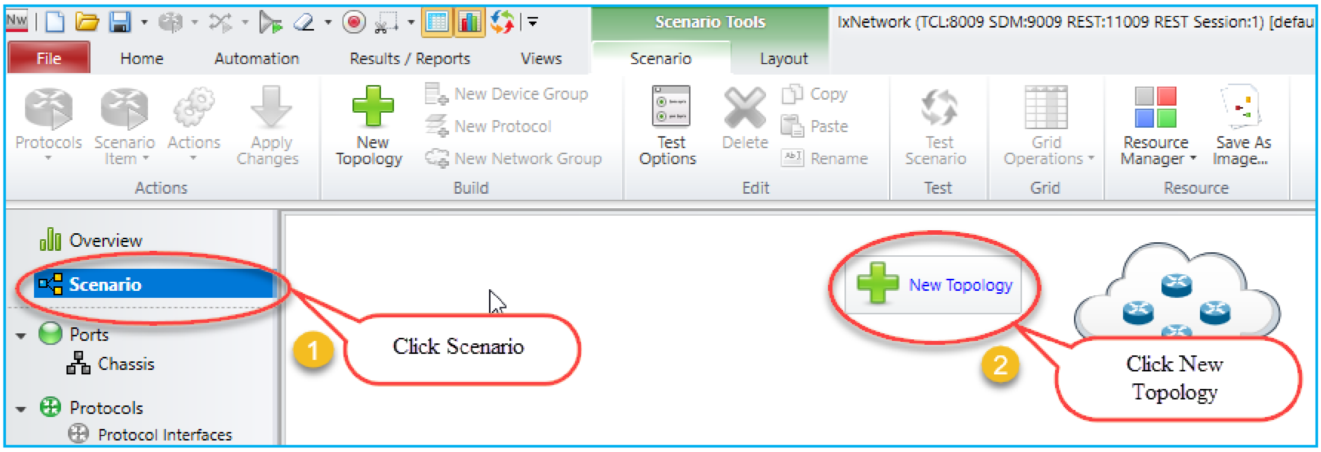
2 Click Add Ports

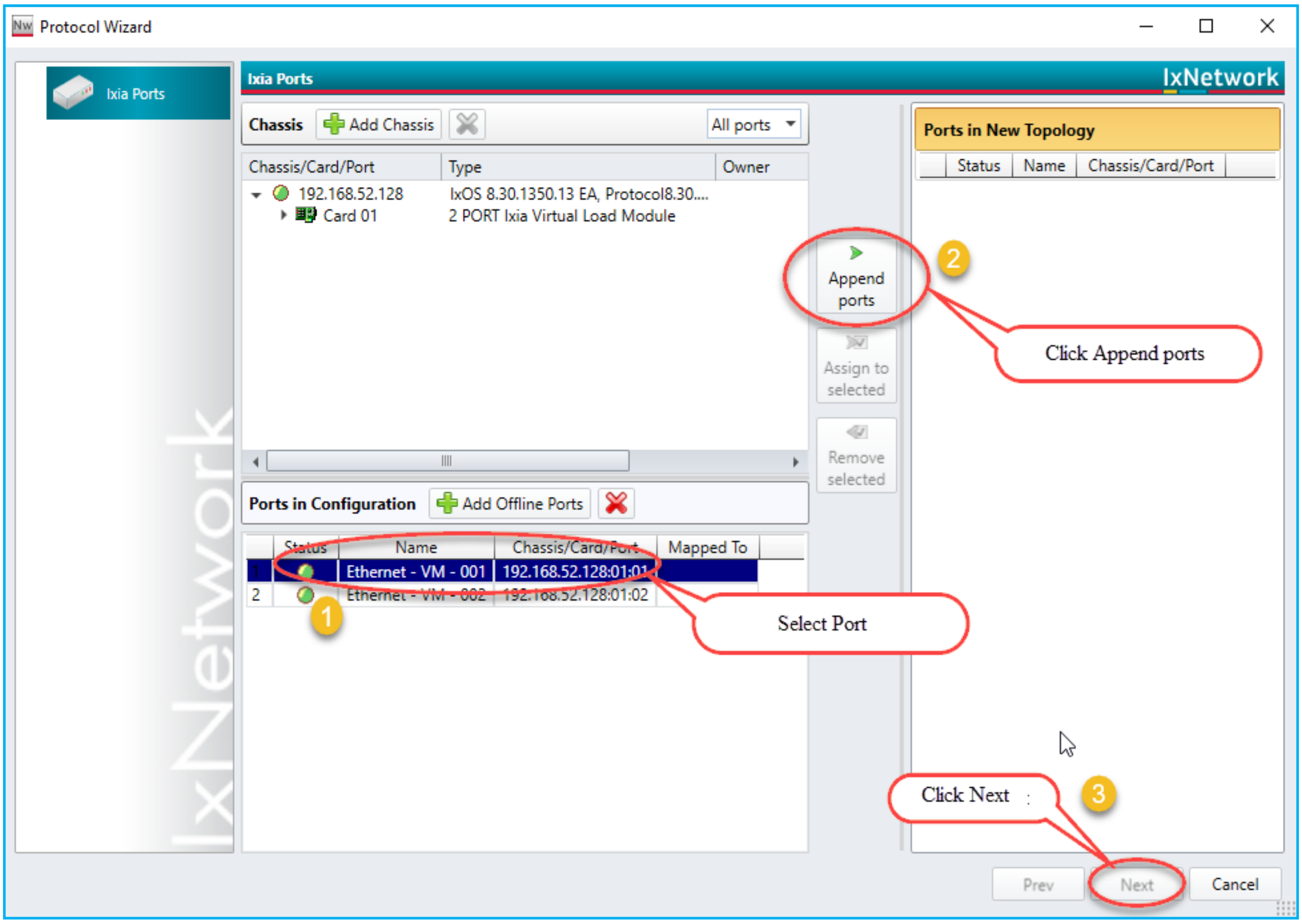
3 Click OK

Fig 2.1.2 Port selection

2.2 Add Topology:

- An **IxNetwork** instance supports one Scenario, which can contain multiple topologies. Each Topology is a collection of one or more test ports. Each port in a Topology is bound to a physical or virtual port and individual ports can be added or removed.





Chassis/Card/Port	Type	Owner
192.168.52.128	IxOS 8.30.1350.13 EA, Protocol8.30....	
Card 01	2 PORT Ixia Virtual Load Module	

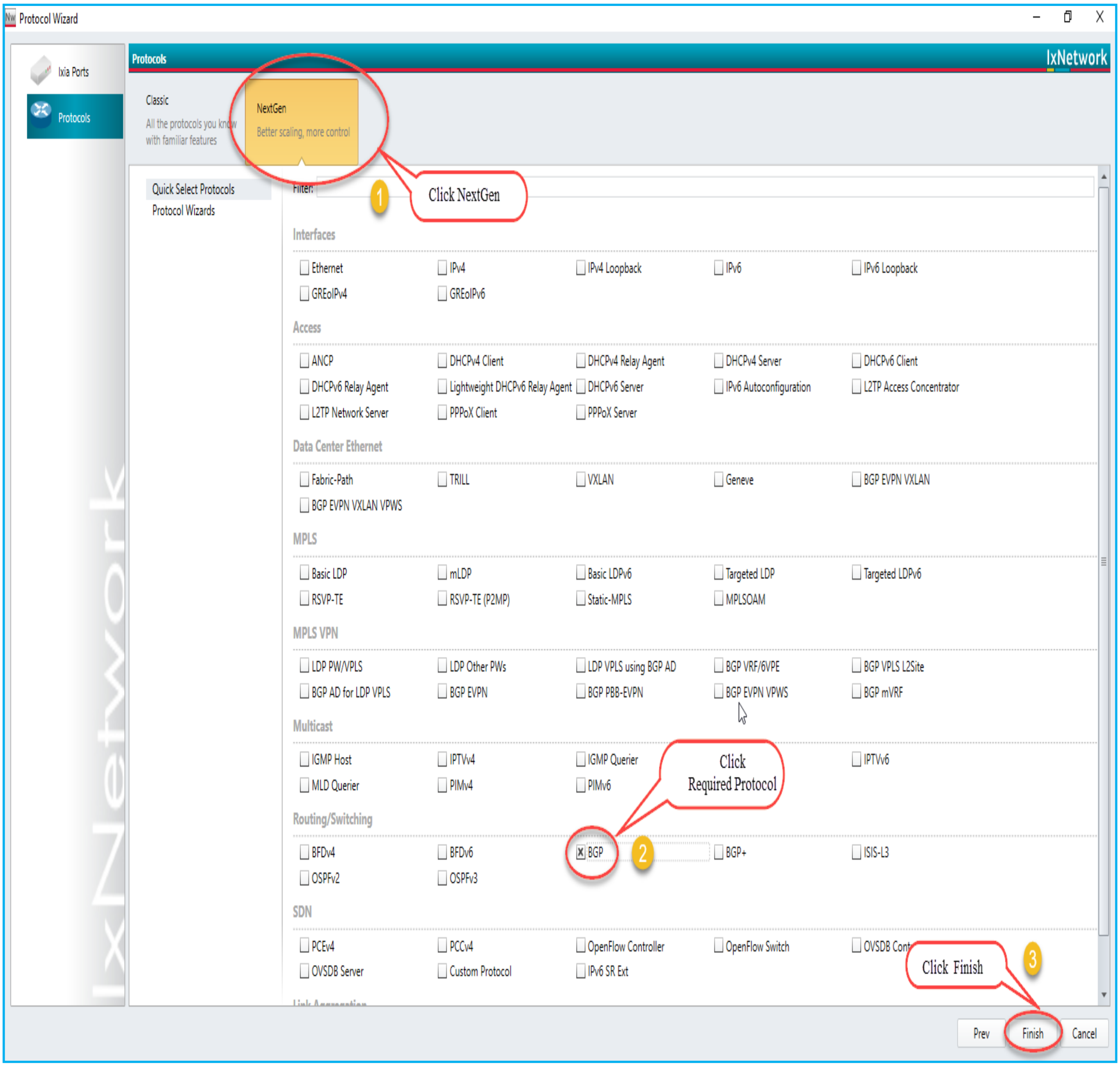
	Status	Name	Chassis/Card/Port	Mapped To
1		Ethernet - VM - 001	192.168.52.128:01:01	
2		Ethernet - VM - 002	192.168.52.128:01:02	

Fig 2.2 Topology with selected ports

2.3 Emulate a Protocol:

- The **Protocols** page in the **Protocol Wizard** allows you to select the protocols in the Topology. The **Protocols** page lists the available **Classic** and **NextGen** protocols under respective tabs. Click **NextGen**, select required protocol for the test.

- Presents all supported protocols in Next Generation Protocol Framework in a single window.



Protocols

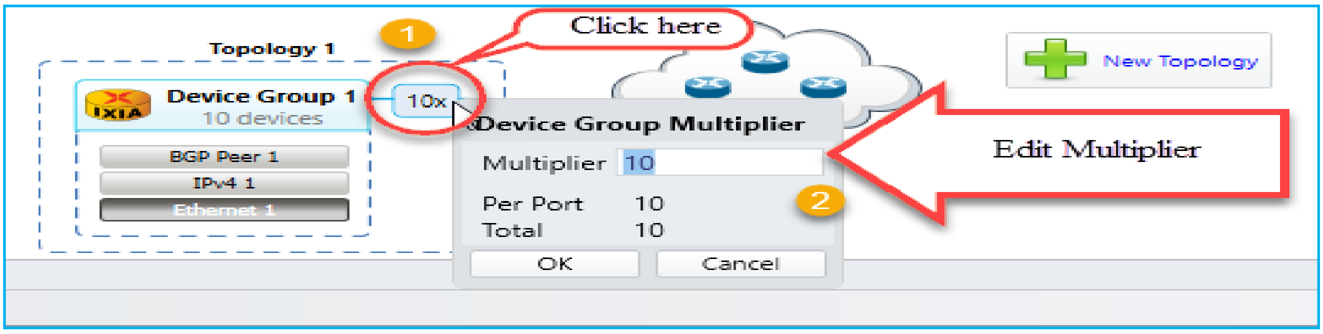
Interfaces				
☐ Ethernet	☐ IPv4	☐ IPv4 Loopback	☐ IPv6	☐ IPv6 Loopback
☐ GREoIPv4	☐ GREoIPv6			
Access				
☐ ANCP	☐ DHCPv4 Client	☐ DHCPv4 Relay Agent	☐ DHCPv4 Server	☐ DHCPv6 Client
☐ DHCPv6 Relay Agent	☐ Lightweight DHCPv6 Relay Agent	☐ DHCPv6 Server	☐ IPv6 Autoconfiguration	☐ L2TP Access Concentrator
☐ L2TP Network Server	☐ PPPoX Client	☐ PPPoX Server		
Data Center Ethernet				
☐ Fabric-Path	☐ TRILL	☐ VXLAN	☐ Geneve	☐ BGP EVPN VXLAN
☐ BGP EVPN VXLAN VPWS				
MPLS				
☐ Basic LDP	☐ mLDP	☐ Basic LDPv6	☐ Targeted LDP	☐ Targeted LDPv6
☐ RSVP-TE	☐ RSVP-TE (P2MP)	☐ Static-MPLS	☐ MPLSOAM	
MPLS VPN				
☐ LDP PW/VPLS	☐ LDP Other PWs	☐ LDP VPLS using BGP AD	☐ BGP VRF/6VPE	☐ BGP VPLS L2Site
☐ BGP AD for LDP VPLS	☐ BGP EVPN	☐ BGP PBB-EVPN	☐ BGP EVPN VPWS	☐ BGP mVRF
Multicast				
☐ IGMP Host	☐ IPTVv4	☐ IGMP Querier		☐ IPTVv6
☐ MLD Querier	☐ PIMv4	☐ PIMv6		
Routing/Switching				
☐ BFDv4	☐ BFDv6	☒ BGP	☐ BGP+	☐ ISIS-L3
☐ OSPFv2	☐ OSPFv3			
SDN				
☐ PCEv4	☐ PCCv4	☐ OpenFlow Controller	☐ OpenFlow Switch	☐ OVSDB Cont
☐ OVSDB Server	☐ Custom Protocol	☐ IPv6 SR Ext		
Link Aggregation				

Fig 2.3 Selected BGP protocol

2.4 Device Group Multiplier:

- A Device Group has similar Devices per test port. A Device can be a router, host, switch, and so on. It can run multiple protocols and protocol stacks.

- A Device Group count is the number of instances in the group. A configuration can be scaled by modeling a group of *n* Devices per test port by changing the multiplier.



Device Group Multiplier

Multiplier 10

Per Port 10

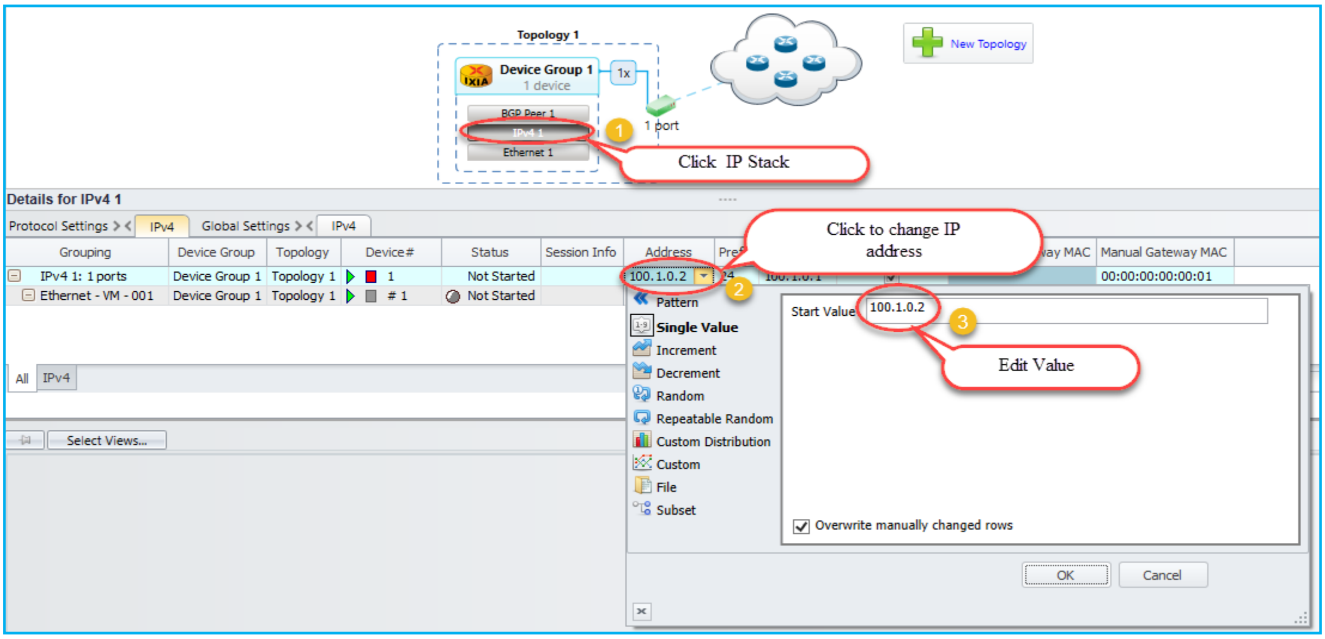
Total 10

OK Cancel

Fig 2.4 Emulate number of devices by using device group multiplier

2.5 Edit Protocol Grid:

- The protocol stack displayed in the Scenario view is interactive. Click on a particular protocol stack, edit the values according to the requirement.



Details for IPv4 1

Protocol Settings > < IPv4 | Global Settings > < IPv4

Grouping	Device Group	Topology	Device#	Status	Session Info	Address	Pre	ay MAC	Manual Gateway MAC
⊟ IPv4 1: 1 ports	Device Group 1	Topology 1	▶ ■ 1	Not Started		100.1.0.2	24		00:00:00:00:00:01
⊟ Ethernet - VM - 001	Device Group 1	Topology 1	▶ ■ # 1	Not Started					

Start Value 100.1.0.2

☑ Overwrite manually changed rows

OK Cancel

Fig 2.5 Configuring interface sections

2.6 Configure BGP:

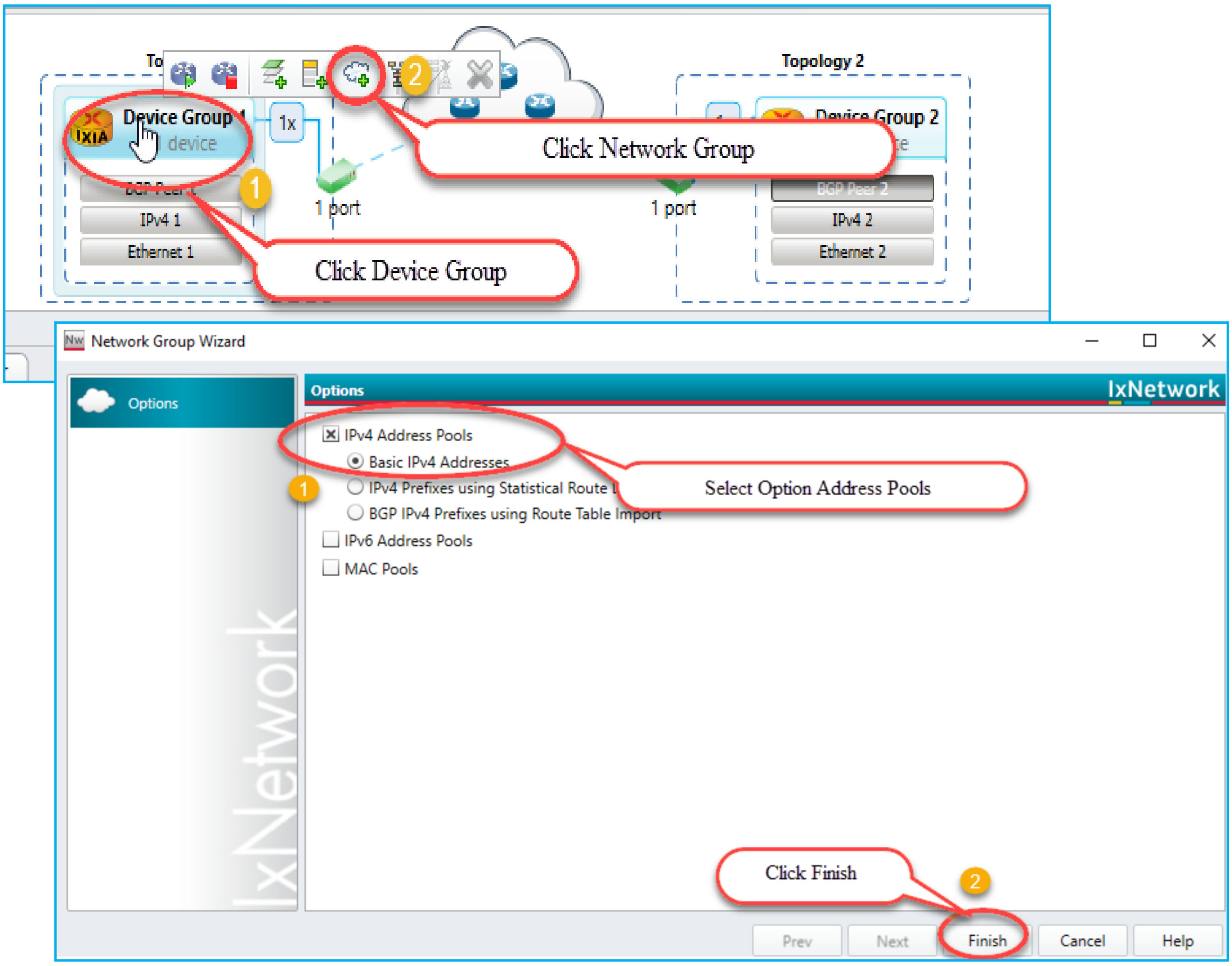
- Configure Interface IP Address to 20.20.20.2 and Gateway Address to 20.20.20.1 in device group 1 IP stack by using method 2.5. Configure Interface IP Address to 20.20.20.1 and Gateway Address to 20.20.20.2 in device group 2 IP stack using by method 2.5

- Similarly configure Local IP to 20.20.20.2 and DUT IP to 20.20.20.1 in BGP Stack in device group 1 by clicking the BGP stack. Configure Local IP to 20.20.20.1 and DUT IP to 20.20.20.2 in BGP Stack in device group 2 by clicking the BGP stack.

2.7 Add Network Group:

- A Network Group represents a set of L3 networks (sub-netted or switched) with optional information explaining the reachability to each of these networks.

- All Devices connected to a Network Group must belong to one of the networks modeled by that Network Group.



1 Click Device Group

2 Click Network Group

1 Select Option Address Pools

☒ IPv4 Address Pools
● Basic IPv4 Addresses
○ IPv4 Prefixes using Statistical Route
○ BGP IPv4 Prefixes using Route Table Import
□ IPv6 Address Pools
□ MAC Pools

2 Click Finish

Fig 2.7 Route Profile addition by using network groups

2.8 Start Protocols:

- ➢ Click **Start All** to start all the protocols configured in the test.

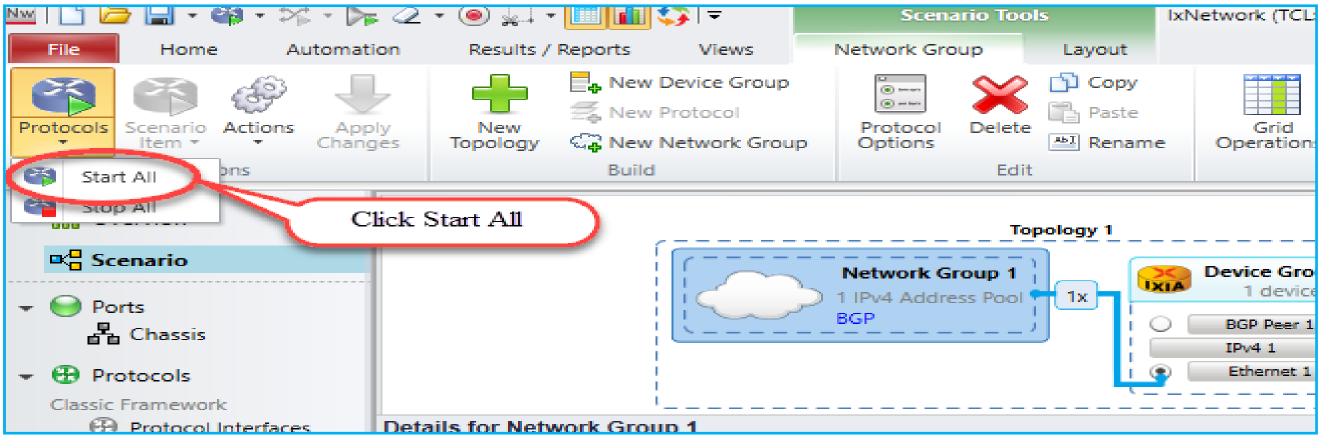


Fig 2.8 Brings up all protocol stacks

2.9 Configure Traffic:

- ➢ The Advanced Traffic Wizard helps to integrate the options for traffic configuration in the control plane and data plane of IxNetwork, thereby facilitating quick setup of large scale testing.

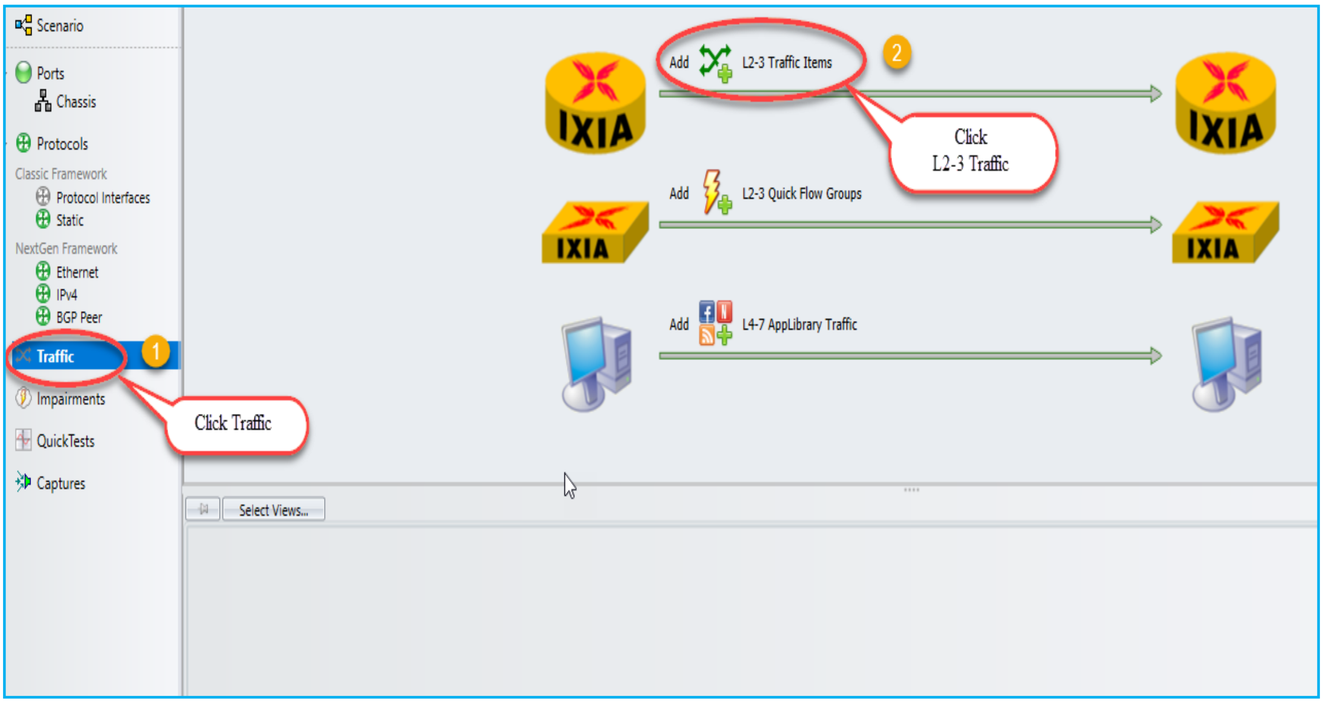
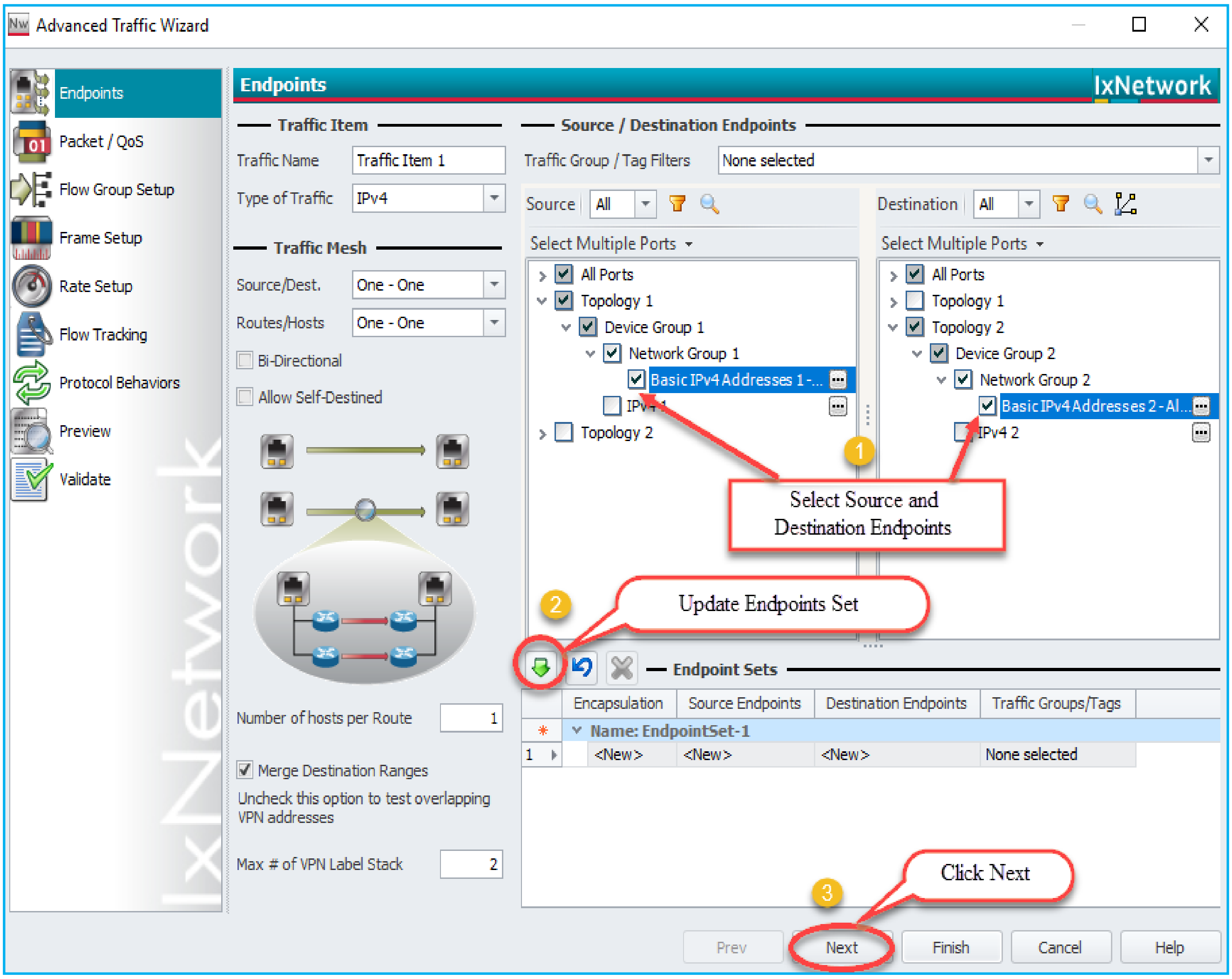


Fig 2.9 Configures L2-3 traffic

2.10 Add Endpoints to Traffic:

- ➢ The Endpoints dialog is the first dialog in a series that form the Advanced Traffic Wizard. To access the Endpoints dialog, click the **Endpoints** tab in the left pane of Advanced Wizard window.

- ➢ The Endpoints dialog opens to display the options for selecting the traffic endpoints.



Advanced Traffic Wizard

Endpoints

Traffic Item

Traffic Name: Traffic Item 1

Type of Traffic: IPv4

Traffic Mesh

Source/Dest.: One - One

Routes/Hosts: One - One

□ Bi-Directional

□ Allow Self-Destined

Number of hosts per Route: 1

☑ Merge Destination Ranges

Uncheck this option to test overlapping VPN addresses

Max # of VPN Label Stack: 2

Source / Destination Endpoints

Traffic Group / Tag Filters: None selected

Source: All

Destination: All

1 Select Source and Destination Endpoints

2 Update Endpoints Set

3 Click Next

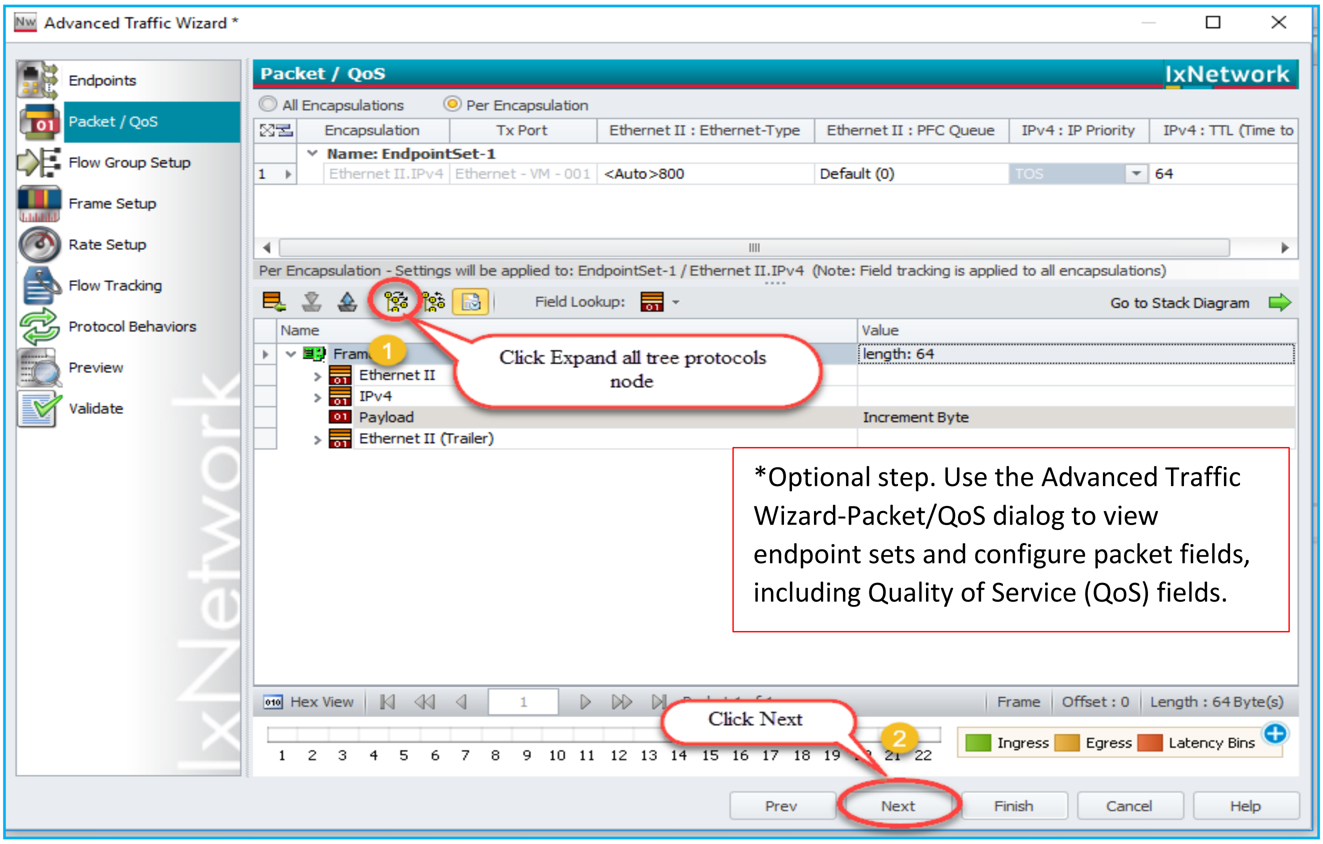
Endpoint Sets

	Encapsulation	Source Endpoints	Destination Endpoints	Traffic Groups/Tags
*	Name: EndpointSet-1			
1	<New>	<New>	<New>	None selected

Fig 2.10 Configures source and destinations endpoints set

2.11 Edit Packet and Setup Flow Groups:

*Editing the packet and setting up flow groups is optional.



Advanced Traffic Wizard *

Packet / QoS

○ All Encapsulations ● Per Encapsulation

	Encapsulation	Tx Port	Ethernet II : Ethernet-Type	Ethernet II : PFC Queue	IPv4 : IP Priority	IPv4 : TTL (Time to
	Name: EndpointSet-1					
1	Ethernet II.IPv4	Ethernet - VM - 001	<Auto>800	Default (0)	TOS	64

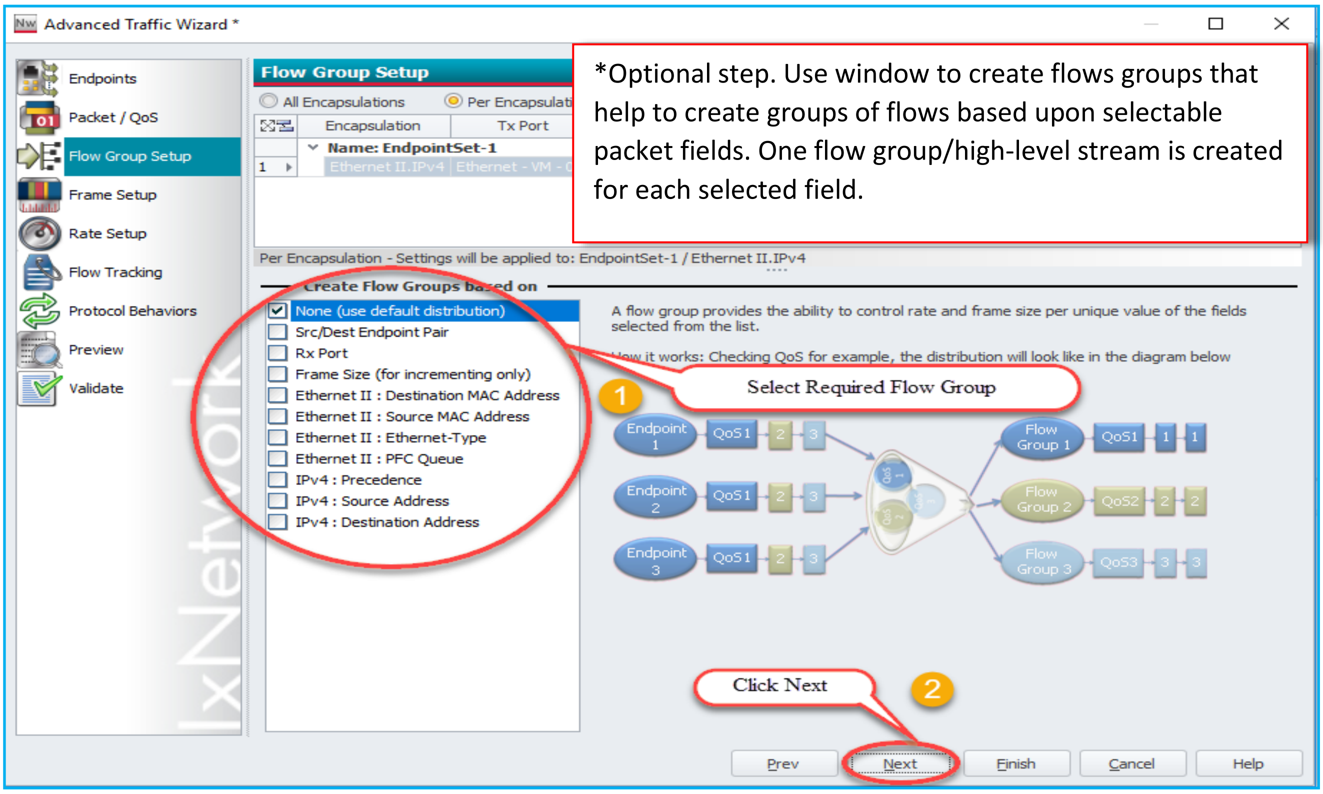
Per Encapsulation - Settings will be applied to: EndpointSet-1 / Ethernet II.IPv4 (Note: Field tracking is applied to all encapsulations)

Name	Value
Frame	length: 64
Ethernet II	
IPv4	
Payload	Increment Byte
Ethernet II (Trailer)	

1 Click Expand all tree protocols node

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

2 Click Next



Advanced Traffic Wizard *

Flow Group Setup

○ All Encapsulations ● Per Encapsulati

	Encapsulation	Tx Port
	Name: EndpointSet-1	
1	Ethernet II.IPv4	Ethernet - VM -

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

Per Encapsulation - Settings will be applied to: EndpointSet-1 / Ethernet II.IPv4

Create Flow Groups based on

- ☑ None (use default distribution)
- □ Src/Dest Endpoint Pair
- □ Rx Port
- □ Frame Size (for incrementing only)
- □ Ethernet II : Destination MAC Address
- □ Ethernet II : Source MAC Address
- □ Ethernet II : Ethernet-Type
- □ Ethernet II : PFC Queue
- □ IPv4 : Precedence
- □ IPv4 : Source Address
- □ IPv4 : Destination Address

A flow group provides the ability to control rate and frame size per unique value of the fields selected from the list.

How it works: Checking QoS for example, the distribution will look like in the diagram below

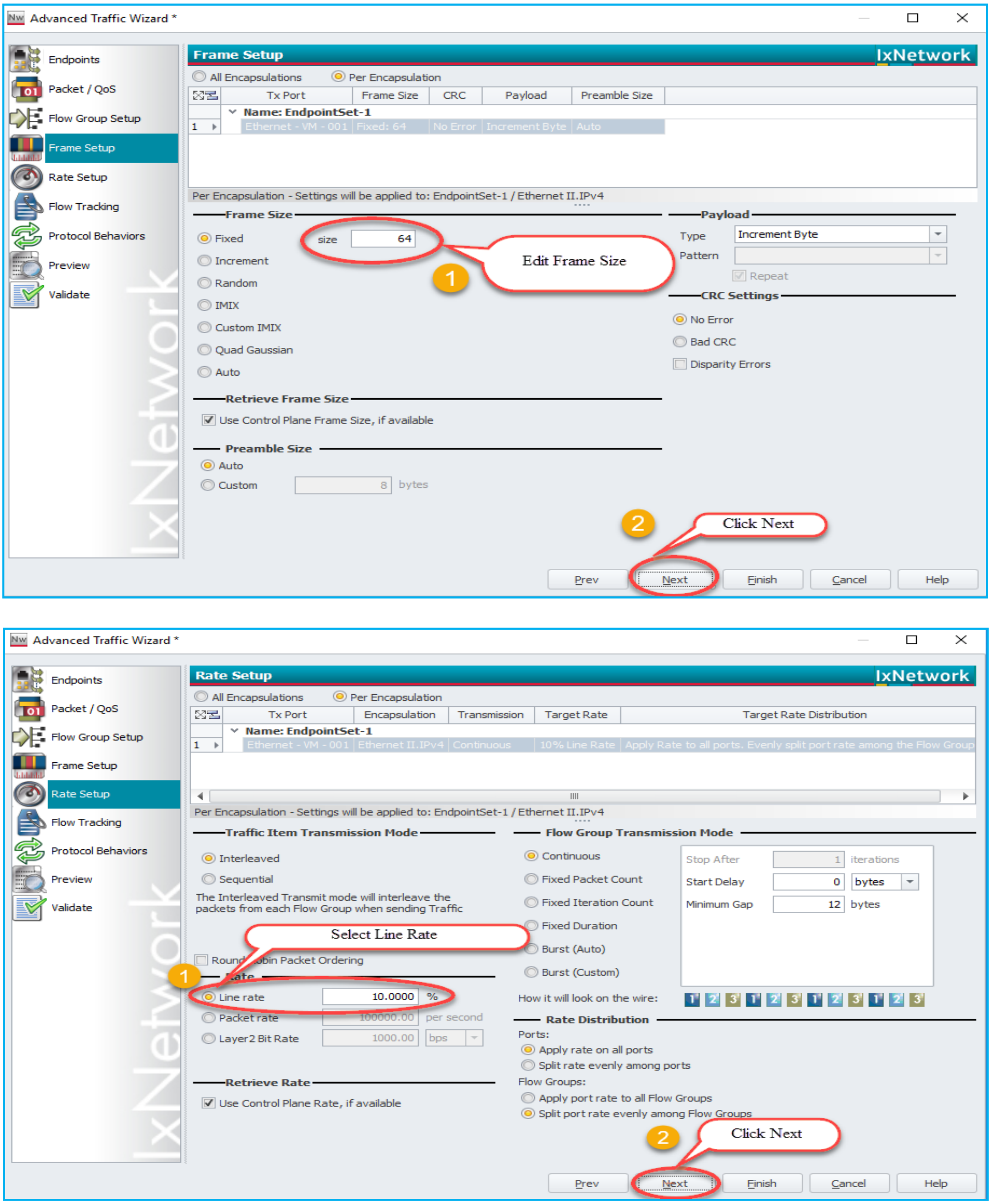
1 Select Required Flow Group

2 Click Next

Fig 2.11 Customizing the packet and creating flow groups

2.12 Setup Frame Size and Rate:

*Setting up the frame Size and Line rate is optional.



Frame Setup

○ All Encapsulations ● Per Encapsulation

Tx Port	Frame Size	CRC	Payload	Preamble Size
Name: EndpointSet-1				
Ethernet - VM - 001	Fixed: 64	No Error	Increment Byte	Auto

Per Encapsulation - Settings will be applied to: EndpointSet-1 / Ethernet II.IPv4

Frame Size: ● Fixed size 64; ○ Increment; ○ Random; ○ IMIX; ○ Custom IMIX; ○ Quad Gaussian; ○ Auto

Payload: Type Increment Byte; Pattern; ☑ Repeat

CRC Settings: ● No Error; ○ Bad CRC; □ Disparity Errors

Retrieve Frame Size: ☑ Use Control Plane Frame Size, if available

Preamble Size: ● Auto; ○ Custom 8 bytes

1 Edit Frame Size

2 Click Next

Rate Setup

○ All Encapsulations ● Per Encapsulation

Tx Port	Encapsulation	Transmission	Target Rate	Target Rate Distribution
Name: EndpointSet-1				
Ethernet - VM - 001	Ethernet II.IPv4	Continuous	10% Line Rate	Apply Rate to all ports. Evenly split port rate among the Flow Group

Per Encapsulation - Settings will be applied to: EndpointSet-1 / Ethernet II.IPv4

Traffic Item Transmission Mode: ● Interleaved; ○ Sequential

The Interleaved Transmit mode will interleave the packets from each Flow Group when sending Traffic

□ Round Robin Packet Ordering

Rate: ● Line rate 10.0000 %; ○ Packet rate 100000.00 per second; ○ Layer2 Bit Rate 1000.00 bps

Retrieve Rate: ☑ Use Control Plane Rate, if available

Flow Group Transmission Mode: ● Continuous; ○ Fixed Packet Count; ○ Fixed Iteration Count; ○ Fixed Duration; ○ Burst (Auto); ○ Burst (Custom)

Stop After 1 iterations; Start Delay 0 bytes; Minimum Gap 12 bytes

How it will look on the wire: 1 2 3 1 2 3 1 2 3 1 2 3

Rate Distribution:
Ports: ● Apply rate on all ports; ○ Split rate evenly among ports
Flow Groups: ○ Apply port rate to all Flow Groups; ● Split port rate evenly among Flow Groups

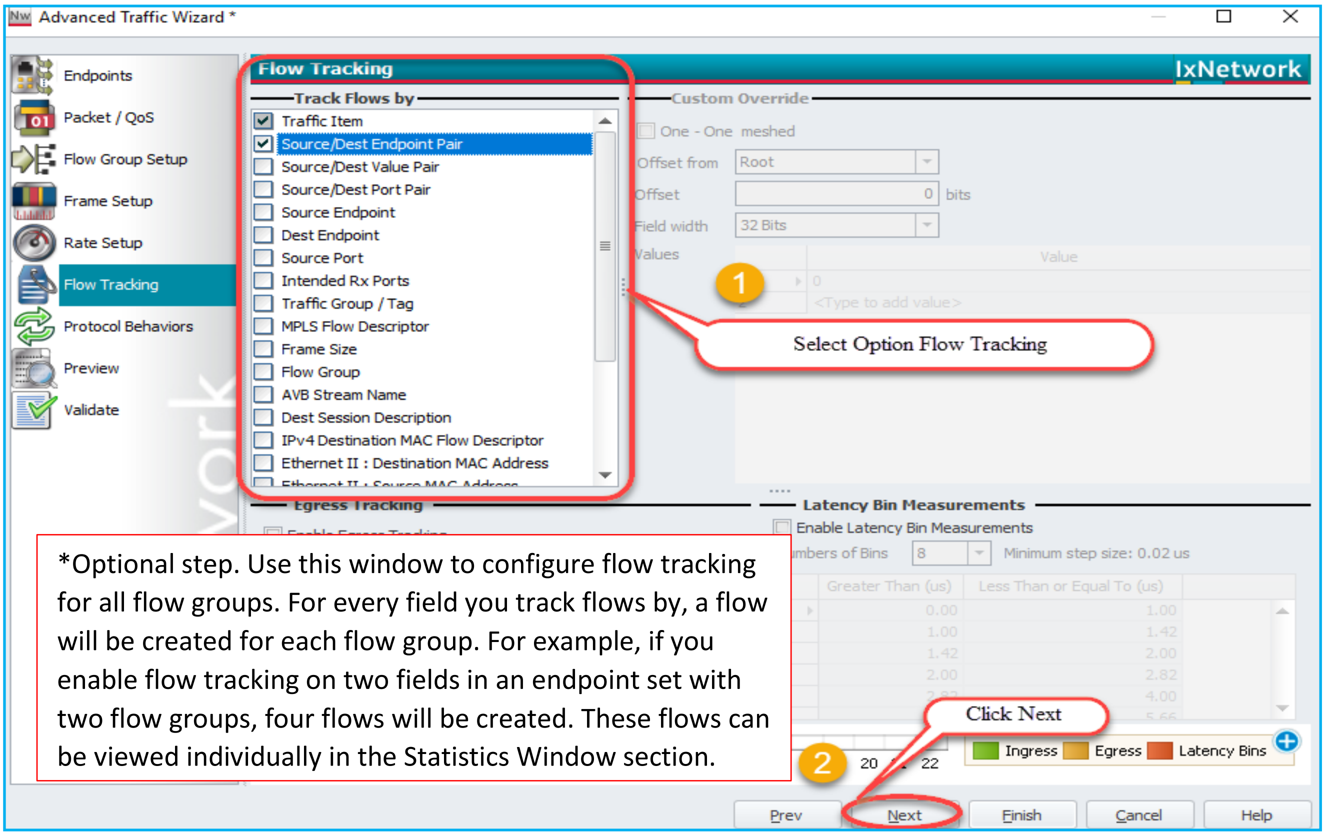
1 Select Line Rate

2 Click Next

Fig 2.12 Setting up the frame size and line rate of the traffic

2.13 Setup Flow Tracking and Protocol Behavior:

*Setting up the flow tracking and Protocol Behavior is optional.



Flow Tracking

Track Flows by
- ☑ Traffic Item
- ☑ Source/Dest Endpoint Pair
- □ Source/Dest Value Pair
- □ Source/Dest Port Pair
- □ Source Endpoint
- □ Dest Endpoint
- □ Source Port
- □ Intended Rx Ports
- □ Traffic Group / Tag
- □ MPLS Flow Descriptor
- □ Frame Size
- □ Flow Group
- □ AVB Stream Name
- □ Dest Session Description
- □ IPv4 Destination MAC Flow Descriptor
- □ Ethernet II : Destination MAC Address

Custom Override: □ One - One meshed; Offset from: Root; Offset: 0 bits; Field width: 32 Bits

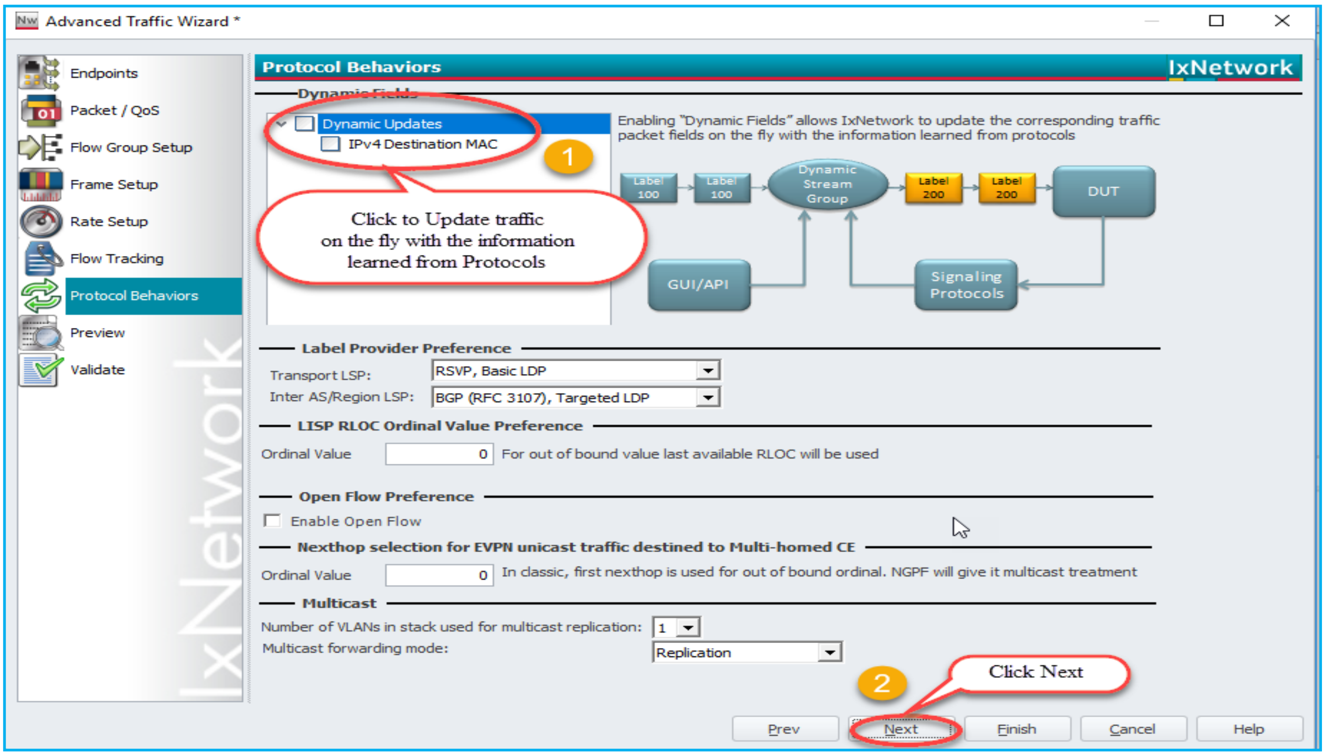
Latency Bin Measurements: □ Enable Latency Bin Measurements; Numbers of Bins: 8; Minimum step size: 0.02 us

Greater Than (us)	Less Than or Equal To (us)
0.00	1.00
1.00	1.42
1.42	2.00
2.00	2.82

1 Select Option Flow Tracking

2 Click Next

*Optional step. Use this window to configure flow tracking for all flow groups. For every field you track flows by, a flow will be created for each flow group. For example, if you enable flow tracking 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.



Protocol Behaviors

Dynamic Fields
- □ Dynamic Updates
  - □ IPv4 Destination MAC

Enabling "Dynamic Fields" allows IxNetwork to update the corresponding traffic packet fields on the fly with the information learned from protocols

1 Click to Update traffic on the fly with the information learned from Protocols

Label Provider Preference
Transport LSP: RSVP, Basic LDP
Inter AS/Region LSP: BGP (RFC 3107), Targeted LDP

LISP RLOC Ordinal Value Preference
Ordinal Value: 0 For out of bound value last available RLOC will be used

Open Flow Preference
□ Enable Open Flow

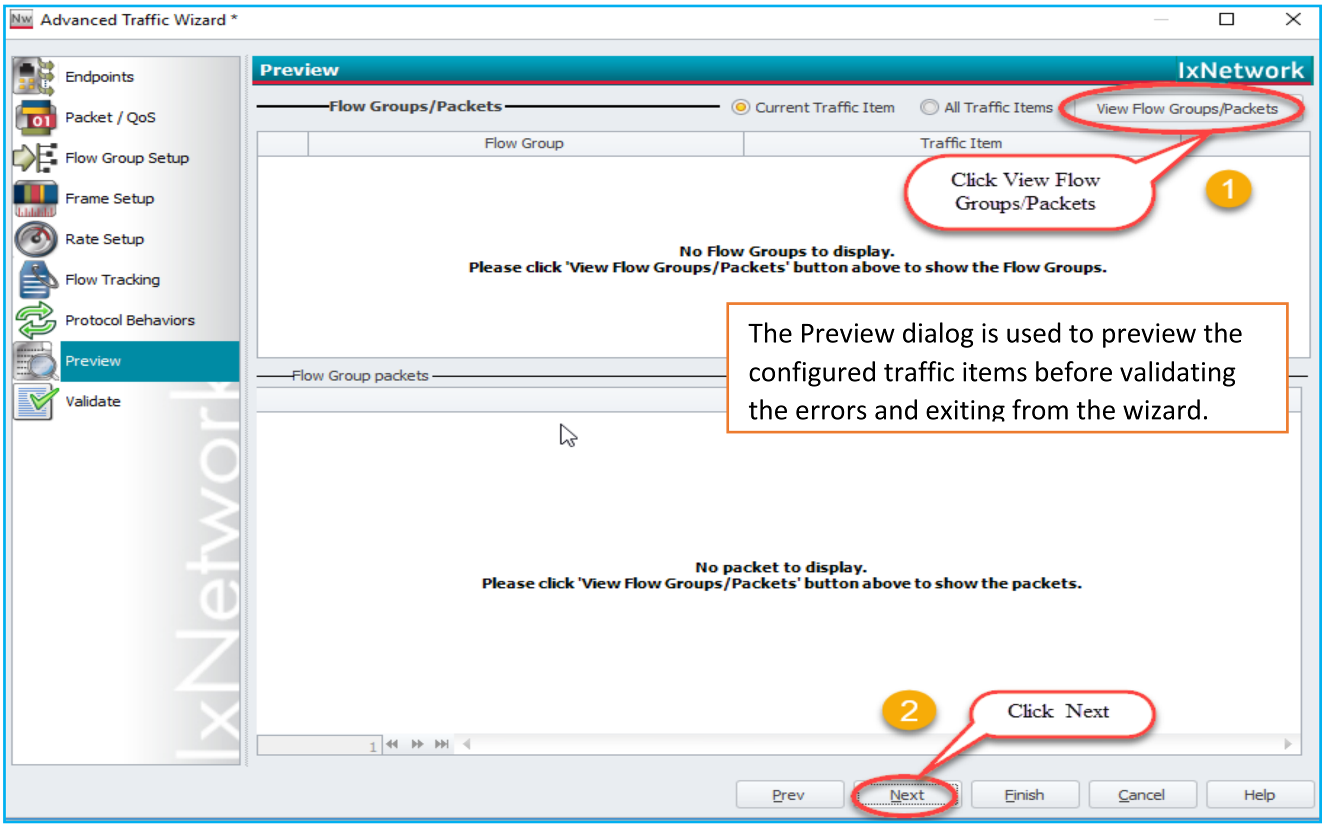
Nexthop selection for EVPN unicast traffic destined to Multi-homed CE
Ordinal Value: 0 In classic, first nexthop is used for out of bound ordinal. NGPF will give it multicast treatment

Multicast
Number of VLANs in stack used for multicast replication: 1
Multicast forwarding mode: Replication

2 Click Next

Fig 2.13 Setting up the flow tracks and traffic update option

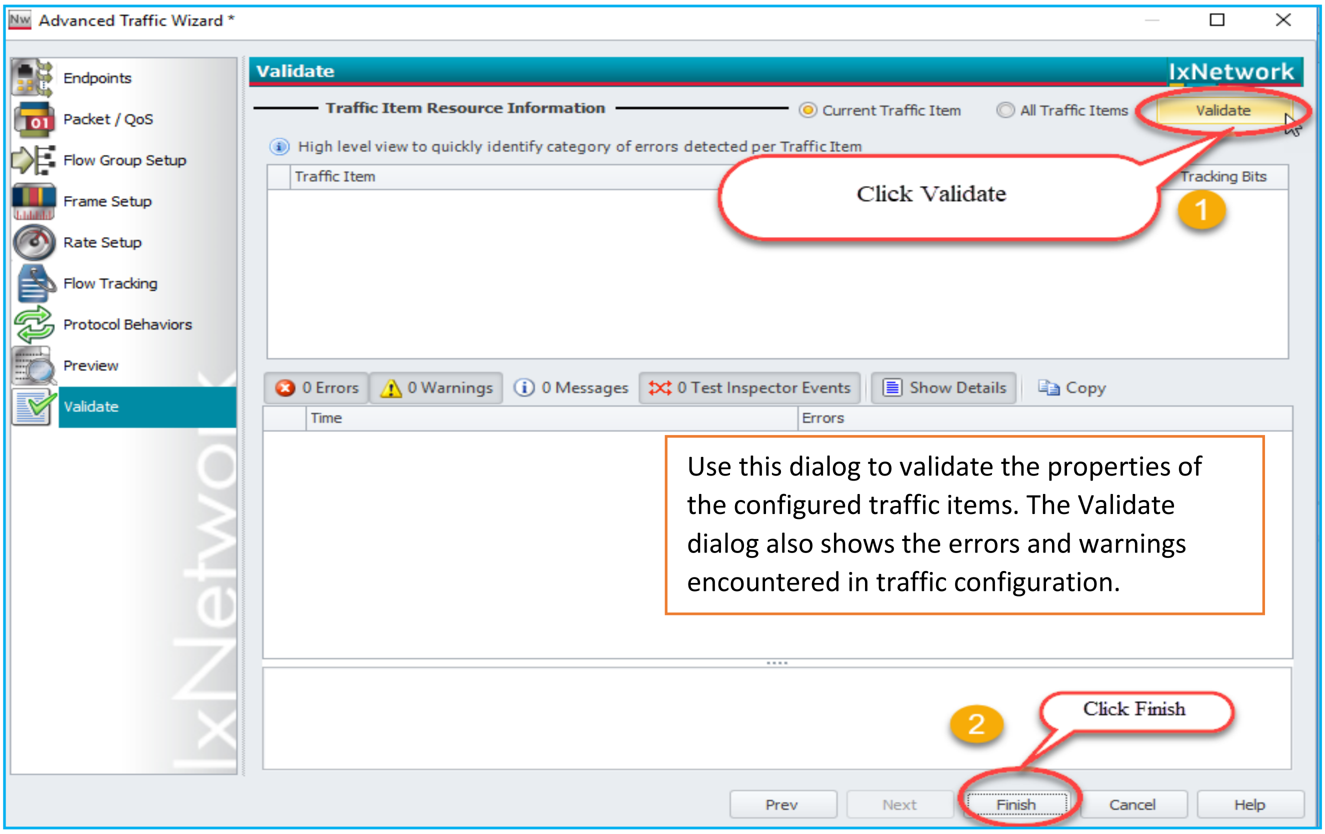
2.14 Validate Traffic:



No Flow Groups to display.
Please click 'View Flow Groups/Packets' button above to show the Flow Groups.

The Preview dialog is used to preview the configured traffic items before validating the errors and exiting from the wizard.

No packet to display.
Please click 'View Flow Groups/Packets' button above to show the packets.

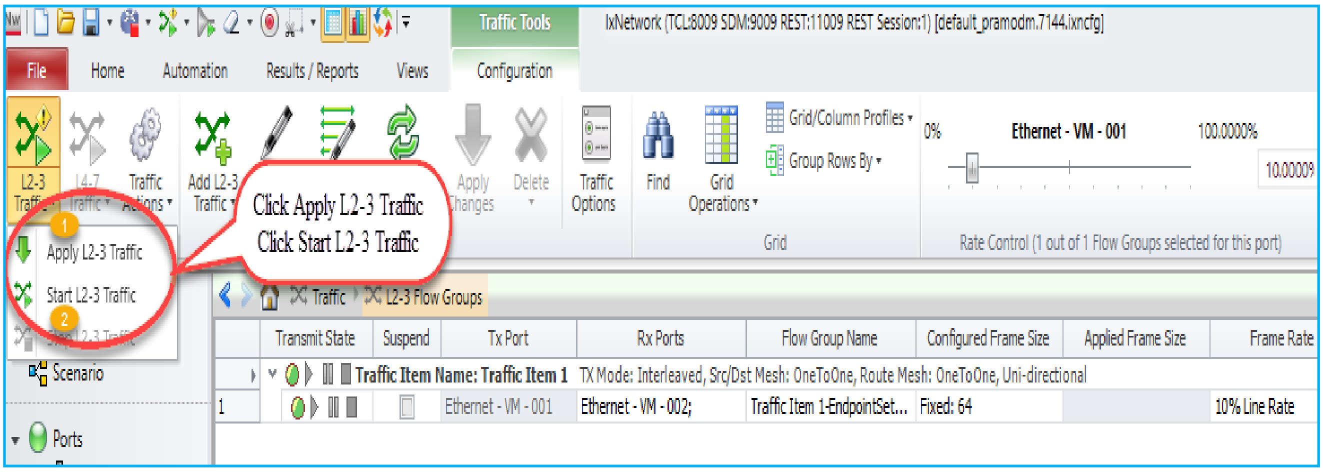


High level view to quickly identify category of errors detected per Traffic Item

Use this dialog to validate the properties of the configured traffic items. The Validate dialog also shows the errors and warnings encountered in traffic configuration.

Fig 2.14 Viewing the flow groups and validating the traffic

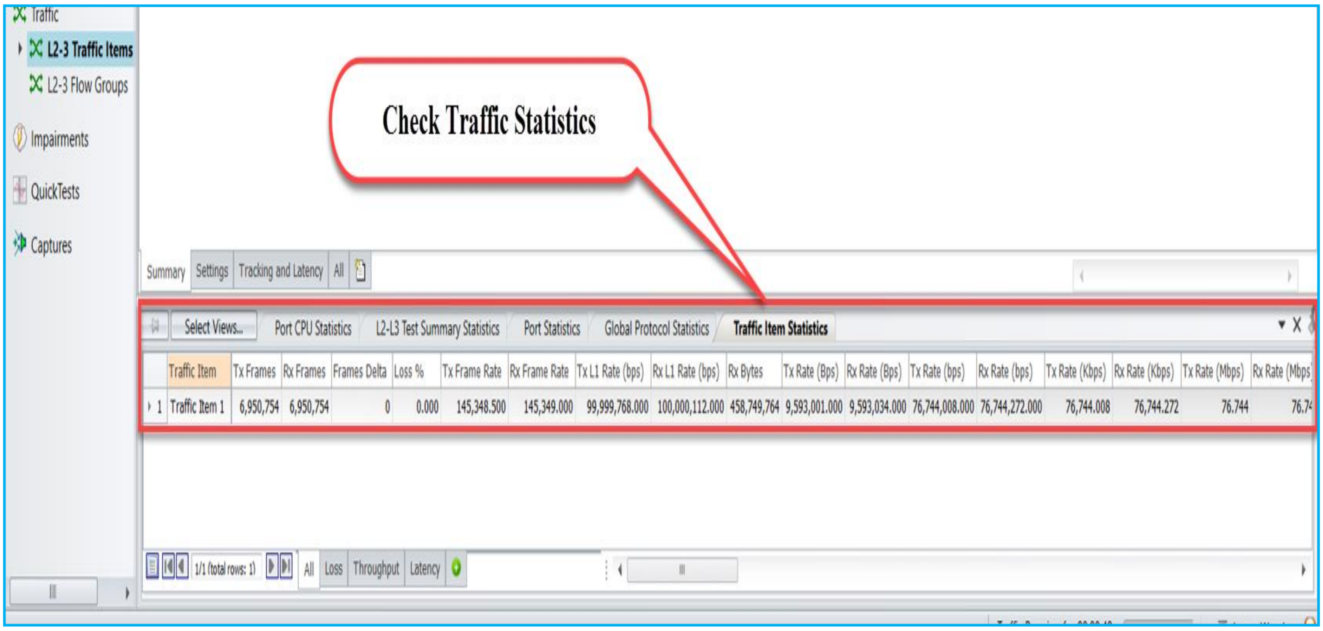
2.15 Apply Traffic, Start Traffic, and Statistics View:



Click Apply L2-3 Traffic
Click Start L2-3 Traffic

	Transmit State	Suspend	Tx Port	Rx Ports	Flow Group Name	Configured Frame Size	Applied Frame Size	Frame Rate
	Traffic Item Name: Traffic Item 1	TX Mode: Interleaved, Src/Dst Mesh: OneToOne, Route Mesh: OneToOne, Uni-directional						
1		□	Ethernet - VM - 001	Ethernet - VM - 002;	Traffic Item 1-EndpointSet...	Fixed: 64		10% Line Rate

Fig 2.15 Applying and Starting traffic



Check Traffic 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 Rate (bps)	Rx Rate (bps)	Tx Rate (Kbps)	Rx Rate (Kbps)	Tx Rate (Mbps)	Rx Rate (Mbps
1	Traffic Item 1	6,950,754	6,950,754	0	0.000	145,348.500	145,349.000	99,999,768.000	100,000,112.000	458,749,764	9,593,001.000	9,593,034.000	76,744,008.000	76,744,272.000	76,744.008	76,744.272	76.744	76.74

Fig 2.15.1 Check for traffic statistics

3. Configure BGP through Automation (HLPyAPI):

This section will walk through to reproduce the same BGP emulation scenario through High Level Python API's to get the user introduced with most of the HLPyAPI's used in NGPF framework.

3.1 Initialize Environment:

Import the Required Packages and Check for the sanity of the System

```
import sys, os
import time, re
sys.path.append('C:/Program Files
(x86)/Ixia/hltapi/8.40.1123.18/TclScripts/lib/hltapi/library/common/ixiangpf/python/')
from ixiatcl import IxiaTcl
from ixiahlt import IxiaHlt
from ixiangpf import IxiaNgpf
from ixiaerror import IxiaError
ixiangpf = IxiaNgpf(ixiahlt)
```

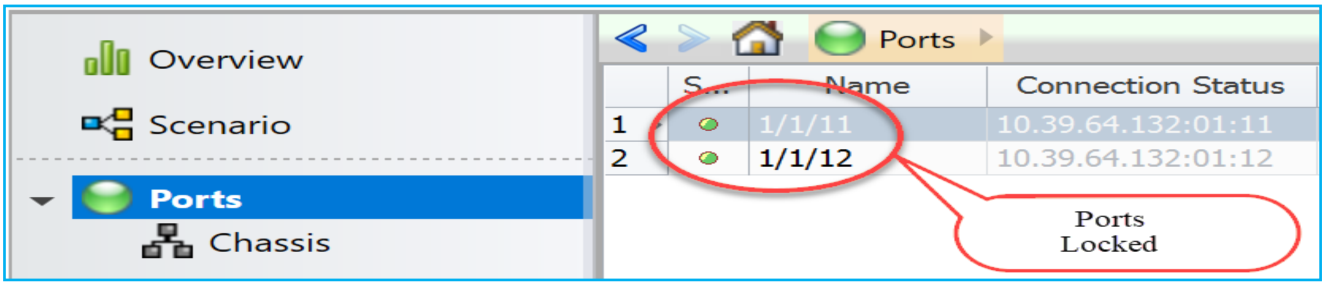
3.2 Add Chassis and Lock Ports:

ixiangpf.connect: Connects to the Ixia chassis with selected ports using the specified port handles

```
chassis_ip                              = "10.39.64.132"
tcl_server                              = "10.39.64.132"
ixnetwork_tcl_server                    = "10.154.163.164:8009"
port_list                               = "1/11 1/12"

connect_result = ixiangpf.connect (
       ixnetwork_tcl_server             = ixnetwork_tcl_server,
       tcl_server                       = tcl_server,
       device                           = chassis_ip,
       port_list                        = port_list,
       break_locks                      = 1,
       reset                            = 1
   )
ports = connect_result['vport_list'].split()
```

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



	S...	Name	Connection Status
1	●	1/1/11	10.39.64.132:01:11
2	●	1/1/12	10.39.64.132:01:12

Ports Locked

Fig 3.2: Chassis connected and selected ports locked

3.3 Create Topology and DeviceGroup:

Ixiangpf.topology_config: Adds Topology to the specified port handle and returns the topology handle and Device Group handle which can be used to configure Device Groups.

```
topologyConfig1 = ixiangpf.topology_config (
    topology_name                                   = """BGP_1 Topology""",
    port_handle                                     = ports[0],
)
topology_1_handle = topologyConfig1['topology_handle']
deviceGroup1 = ixiangpf.topology_config (
    topology_handle                                 = topology_1_handle,
    device_group_name                               = """BGP_1 Device Group""",
    device_group_multiplier                         = "1",
    device_group_enabled                            = "1",
)
deviceGroup_1_handle = deviceGroup1['device_group_handle']

topologyConfig2 = ixiangpf.topology_config (
    topology_name                                   = """BGP_2 Topology""",
    port_handle                                     = ports[1],
)
topology_2_handle = topologyConfig2['topology_handle']
deviceGroup2 = ixiangpf.topology_config (
    topology_handle                                 = topology_2_handle,
    device_group_name                               = """BGP_2 Device Group""",
    device_group_multiplier                         = "1",
    device_group_enabled                            = "1",
)
deviceGroup_2_handle = deviceGroup2['device_group_handle']
```

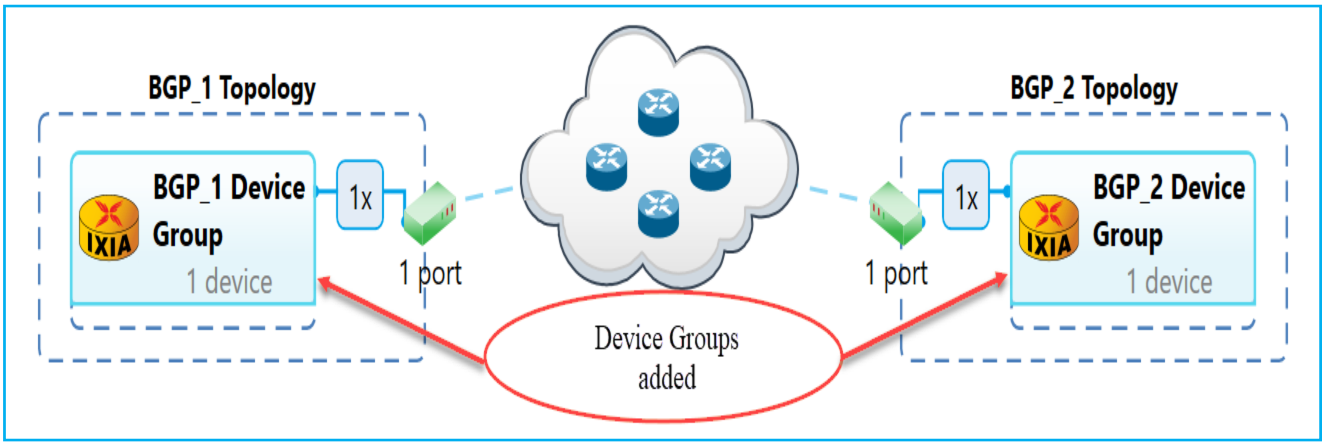


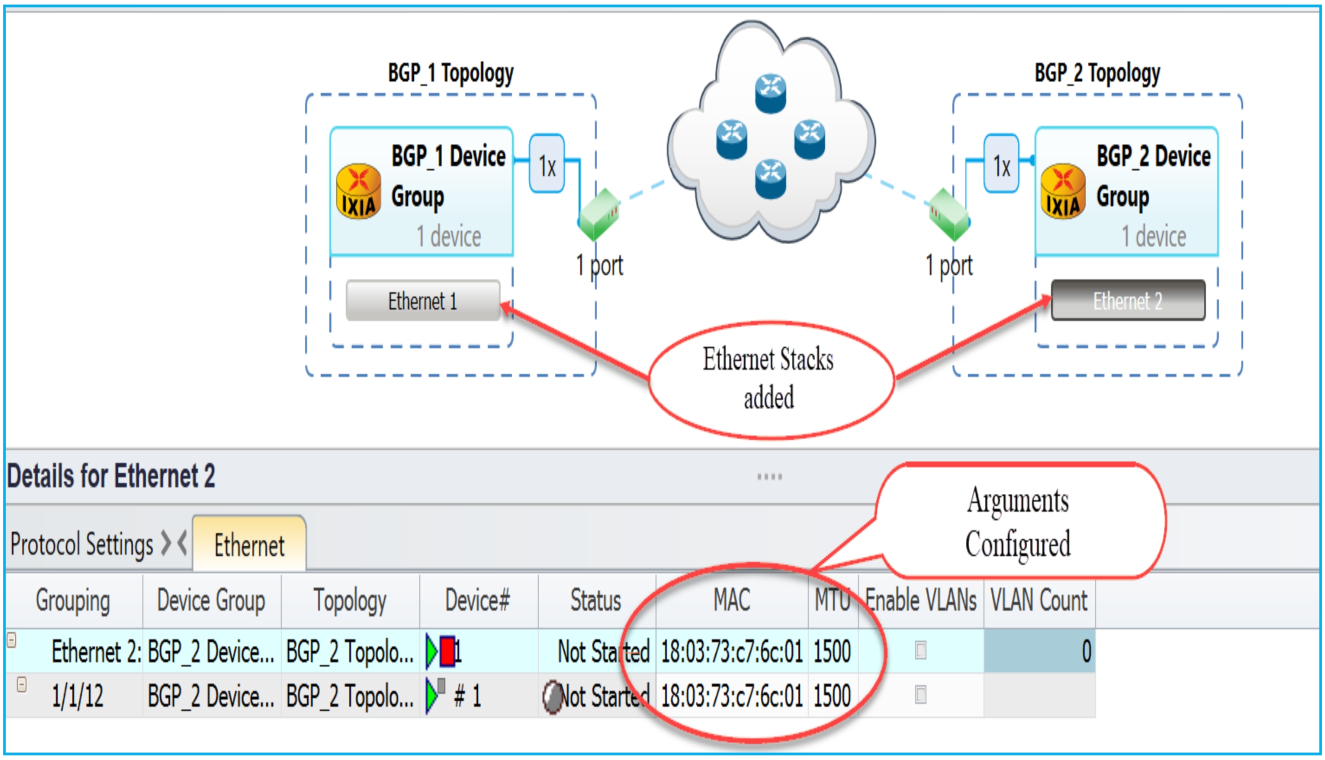
Fig 3.3: Device groups added to respective topologies

3.4 Create Ethernet Stack:

Ixiangpf.interface_config: Configures the protocol stack with the Specified Options by using the Device Group Handle and returns the particular protocol stack handle

```
interfaceConfig1 = ixiangpf.interface_config (
    protocol_name                              = """Ethernet 1""",
    protocol_handle                            = deviceGroup_1_handle,
    mtu                                        = "1500",
    src_mac_addr                               = "18.03.73.c7.6c.b1",
    src_mac_addr_step                          = "00.00.00.00.00.00",
)
ethernet_1_handle = interfaceConfig1['ethernet_handle']

interfaceConfig2 = ixiangpf.interface_config (
    protocol_name                              = """Ethernet 2""",
    protocol_handle                            = deviceGroup_2_handle,
    mtu                                        = "1500",
    src_mac_addr                               = "18.03.73.c7.6c.01",
    src_mac_addr_step                          = "00.00.00.00.00.00",
)
ethernet_2_handle = interfaceConfig2['ethernet_handle']
```



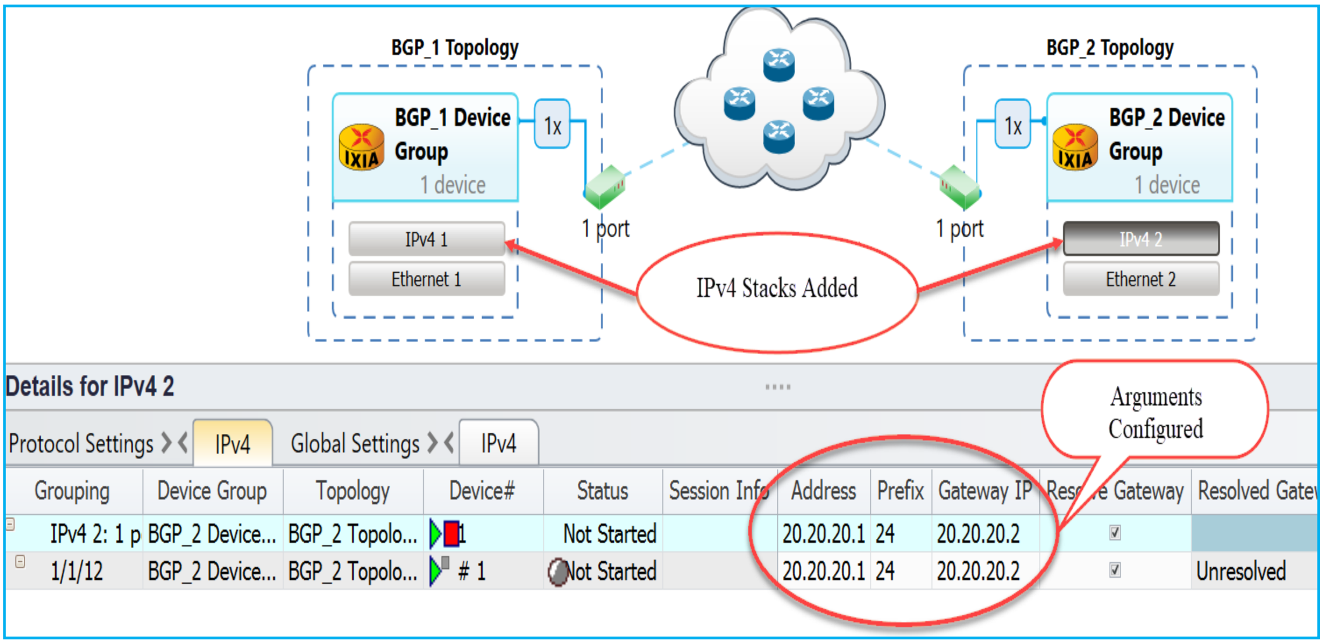
	Grouping	Device Group	Topology	Device#	Status	MAC	MTU	Enable VLANs	VLAN Count
⊟	Ethernet 2:	BGP_2 Device...	BGP_2 Topolo...	1	Not Started	18:03:73:c7:6c:01	1500	□	0
⊟	1/1/12	BGP_2 Device...	BGP_2 Topolo...	# 1	Not Started	18:03:73:c7:6c:01	1500	□	

Fig 3.4: Ethernet stacks added to device groups

3.5 Create Ipv4 Stack:

```
ipv4config1 = ixiangpf.interface_config (
    protocol_name                        = """IPv4 1""",
    protocol_handle                      = ethernet_1_handle,
    ipv4_resolve_gateway                 = "1",
    ipv4_manual_gateway_mac              = "00.00.00.00.00.01",
    gateway                              = "20.20.20.1",
    gateway_step                         = "0.0.0.0",
    intf_ip_addr                         = "20.20.20.2",
    intf_ip_addr_step                    = "0.0.0.0",
    netmask                              = "255.255.255.0",
)
ipv4_1_handle = ipv4config1['ipv4_handle']

ipv4config2 = ixiangpf.interface_config (
    protocol_name                        = """IPv4 2""",
    protocol_handle                      = ethernet_2_handle,
    ipv4_resolve_gateway                 = "1",
    ipv4_manual_gateway_mac              = "00.00.00.00.00.01",
    gateway                              = "20.20.20.2",
    gateway_step                         = "0.0.0.0",
    intf_ip_addr                         = "20.20.20.1",
    intf_ip_addr_step                    = "0.0.0.0",
    netmask                              = "255.255.255.0",
)
ipv4_2_handle = ipv4config2['ipv4_handle']
```



IPv4 Stacks Added

Arguments Configured

Details for IPv4 2

Grouping	Device Group	Topology	Device#	Status	Session Info	Address	Prefix	Gateway IP	Resolve Gateway	Resolved Gatew
IPv4 2: 1 p	BGP_2 Device...	BGP_2 Topolo...	1	Not Started		20.20.20.1	24	20.20.20.2	☑	
1/1/12	BGP_2 Device...	BGP_2 Topolo...	# 1	Not Started		20.20.20.1	24	20.20.20.2	☑	Unresolved

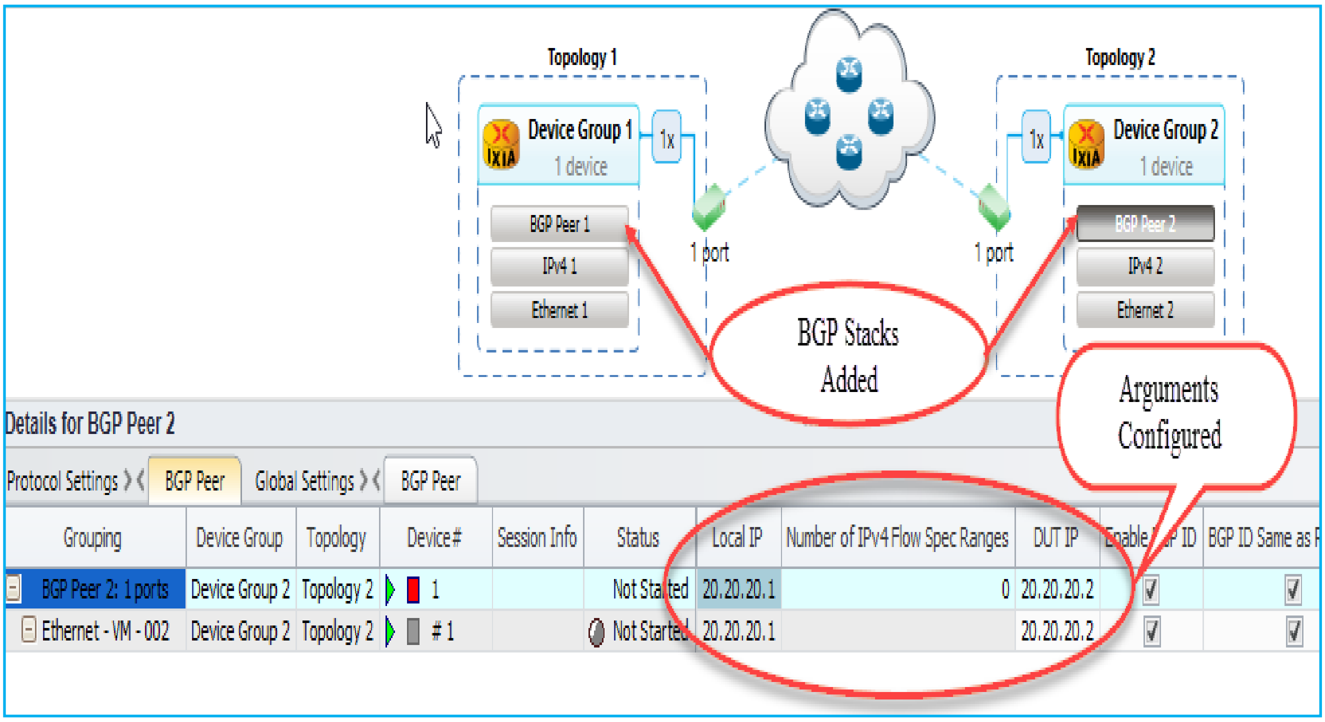
Fig 3.5: IPv4 stacks added to ethernet stacks

3.6 Create BGP:

Ixiangpf.emulation_bgp_config: Configures BGP stack with the specified options by using IPv4 handle and returns the BGP Stack handle

```
 bgpConfig1 = ixiangpf.emulation_bgp_config (
   mode                                        = "enable",
   active                                      = "1",
   handle                                      = ipv4_1_handle,
   remote_ip_addr                              = "20.20.20.1",
)
bgpIpv4Peer_1_handle = bgpConfig1['bgp_handle']

bgpConfig2 = ixiangpf.emulation_bgp_config (
   mode                                        = "enable",
   active                                      = "1",
   handle                                      = ipv4_2_handle,
   remote_ip_addr                              = "20.20.20.2",
)
bgpIpv4Peer_2_handle = bgpConfig2['bgp_handle']
```



Details for BGP Peer 2

Protocol Settings › ‹ BGP Peer | Global Settings › ‹ BGP Peer

Grouping	Device Group	Topology	Device#	Session Info	Status	Local IP	Number of IPv4 Flow Spec Ranges	DUT IP	Enable BGP ID	BGP ID Same as R
⊟ BGP Peer 2: 1 ports	Device Group 2	Topology 2	▶ 🟥 1		Not Started	20.20.20.1	0	20.20.20.2	☑	☑
⊟ Ethernet - VM - 002	Device Group 2	Topology 2	▶ ▮ # 1		● Not Started	20.20.20.1		20.20.20.2	☑	☑

Fig 3.6: BGP stacks added to ipv4 stacks

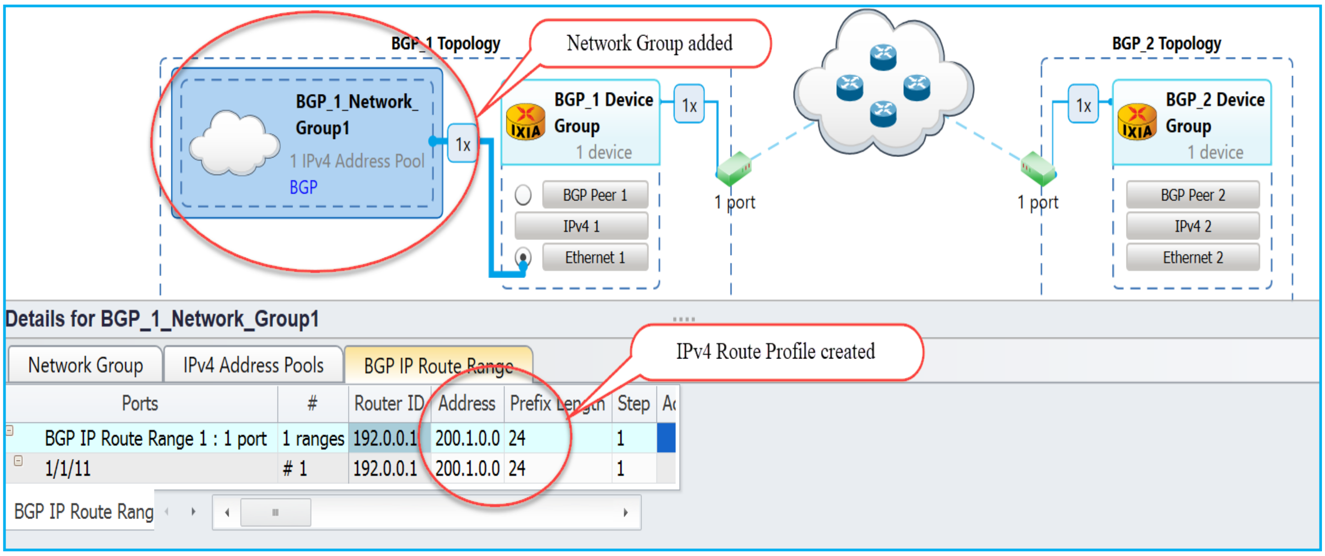
3.7 Create Network Group:

Ixiangpf.multivalue_config: Configures multivalue with specified options by using Device Group handle and topology handle and returns the multivalue handle

```
multiValueConfig1 = ixiangpf.multivalue_config (

    pattern                                      = "counter",
    counter_start                                = "200.1.0.0",
    counter_step                                 = "0.1.0.0",
    nest_step                                    = "0.0.0.1,0.1.0.0",
    nest_owner                                   = '%s,%s' %      (deviceGroup_1_handle
                                                                   topology_1_handle),
    nest_enabled                                 = "0,1",
)
multivalue_4_handle = multiValueConfig1['multivalue_handle']


networkGroupConfig1 = ixiangpf.network_group_config (
    protocol_handle                              = deviceGroup_1_handle,
    protocol_name                                = "BGP_1_Network_Group1",
    multiplier                                   = "1",
    enable_device                                = "1",
    connected_to_handle                          = ethernet_1_handle,
    type                                         = "ipv4-prefix",
    ipv4_prefix_network_address                  = multivalue_4_handle,
    ipv4_prefix_length                           = "24",
)
  networkGroup_1_handle = networkGroupConfig1['network_group_handle’]
```



Details for BGP_1_Network_Group1

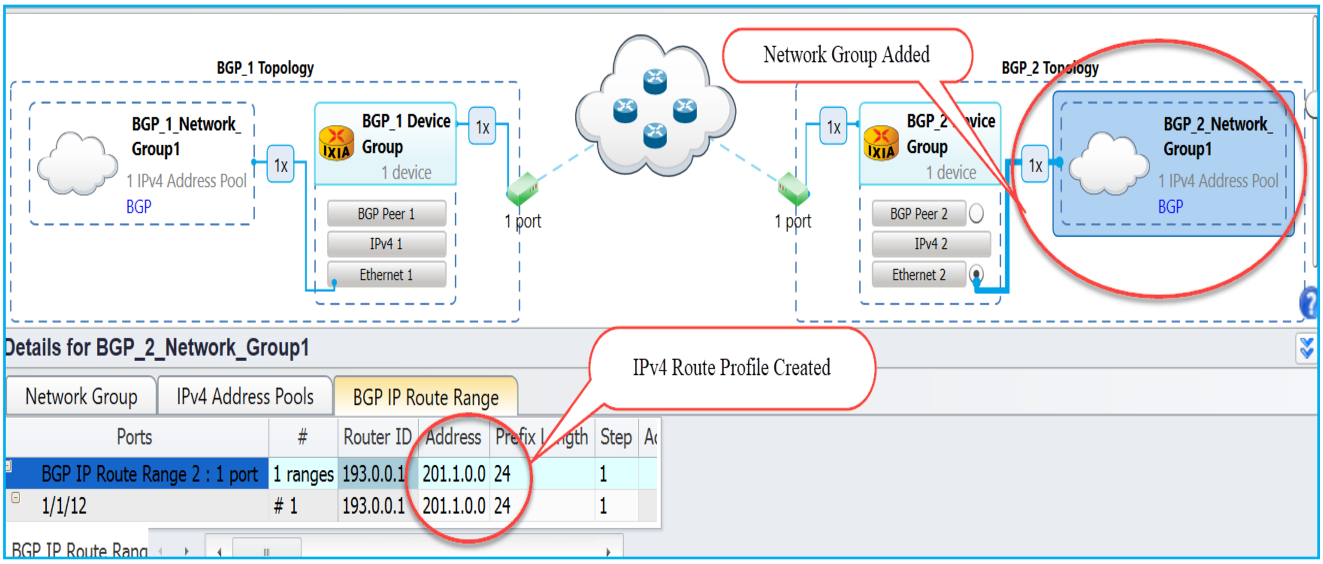
Ports	#	Router ID	Address	Prefix Length	Step	Ac
BGP IP Route Range 1 : 1 port	1 ranges	192.0.0.1	200.1.0.0	24	1	
1/1/11	# 1	192.0.0.1	200.1.0.0	24	1	

Fig 3.7 Adding BGP network group to device group 1

Ixiangpf.network_group_config: Configures Network Group with specified options by using Device Group handle, topology handle, ethernet handle and returns the Network Group handle

```
multiValueConfig2 = ixiangpf.multivalue_config (
    pattern                                         = "counter",
    counter_start                                   = "201.1.0.0",
    counter_step                                    = "0.1.0.0",
    counter_direction                               = "increment",
    nest_step                                       = "0.0.0.1,0.1.0.0",
    nest_owner                                      = '%s,%s' % (deviceGroup_2_handle,
                                                                 topology_2_handle),
    nest_enabled                                    = "0,1",
)
multivalue_10_handle = multiValueConfig2['multivalue_handle']

networkGroupConfig2 = ixiangpf.network_group_config (
    protocol_handle                                 = deviceGroup_2_handle,
    protocol_name                                   = "BGP_2_Network_Group1",
    multiplier                                      = "1",
    enable_device                                   = "1",
    connected_to_handle                             = ethernet_2_handle,
    type                                            = "ipv4-prefix",
    ipv4_prefix_network_address                     = multivalue_10_handle,
    ipv4_prefix_length                              = "24",
)
networkGroup_3_handle = networkGroupConfig2['network_group_handle']
```



Details for BGP_2_Network_Group1

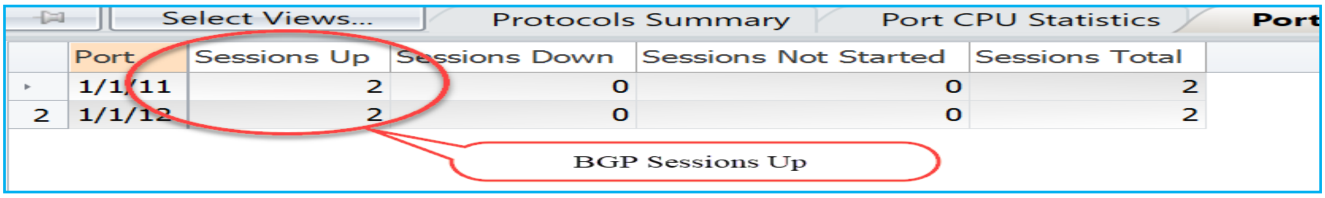
Ports	#	Router ID	Address	Prefix Length	Step	Ac
BGP IP Route Range 2 : 1 port	1 ranges	193.0.0.1	201.1.0.0	24	1	
1/1/12	# 1	193.0.0.1	201.1.0.0	24	1	

Fig 3.7.1 Adding BGP network group to device group 2

3.8 Start Protocols:

ixiangpf.test_control: Start/Stop all the protocols configured in the test session

```
testControl = ixiangpf.test_control (action='start_all_protocols')
print("Waiting for 45 seconds for the protocols to converge")
timer = 30
time.sleep(timer)
```



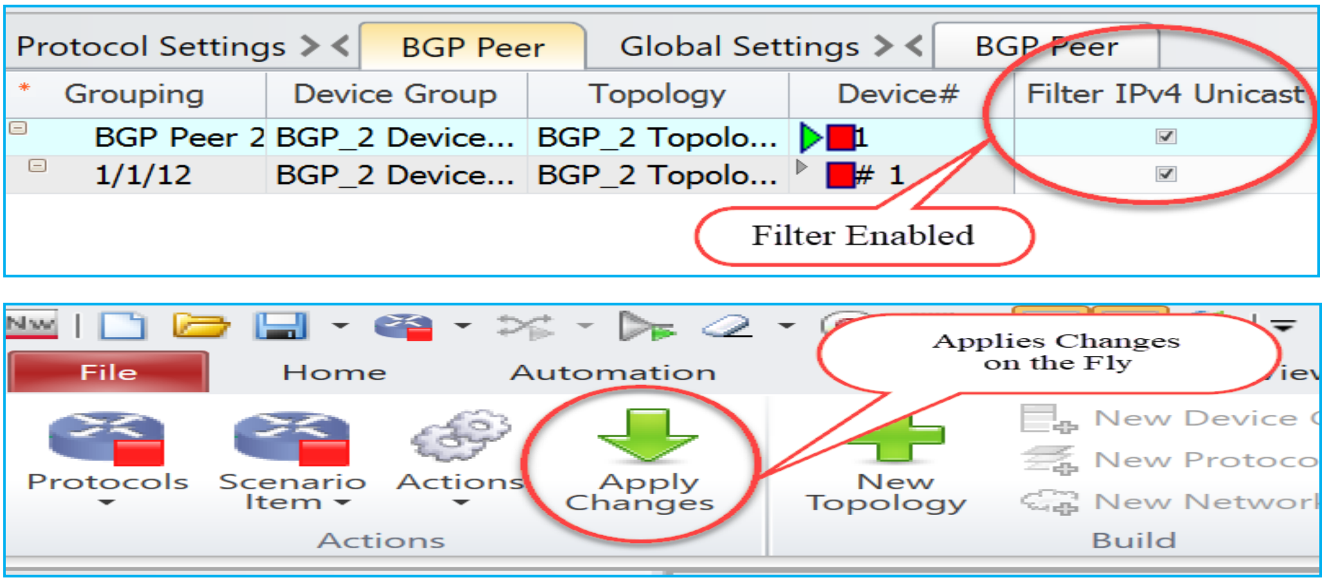
	Port	Sessions Up	Sessions Down	Sessions Not Started	Sessions Total
▸	1/1/11	2	0	0	2
2	1/1/12	2	0	0	2

Fig 3.8 Starting all protocol stacks to come up

3.9 Enable Filter and Apply Changes on the Fly:

```
bgp_1_status = ixiangpf.emulation_bgp_config (

    handle                                    = bgpIpv4Peer_1_handle,
    mode                                      = 'modify',
    ipv4_filter_unicast_nlri                  = '1',
)
applyChanges = ixiangpf.test_control (
    handle                                    = ipv4_1_handle,
    action                                    = 'apply_on_the_fly_changes',
)
```



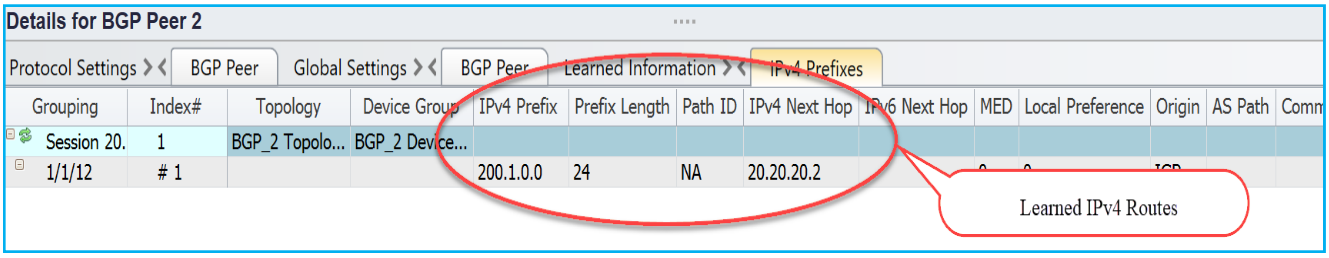
Grouping	Device Group	Topology	Device#	Filter IPv4 Unicast
BGP Peer 2	BGP_2 Device...	BGP_2 Topolo...	1	☑
1/1/12	BGP_2 Device...	BGP_2 Topolo...	# 1	☑

Fig 3.9 Enabling the route filter and applying the changes on the fly

3.10 Retrieve Learned Info:

```
bgpLearnedInfo = ixiangpf.emulation_bgp_info (
    handle                                    = bgpIpv4Peer_1_handle,
    mode                                      = 'learned_info');
pprint(bgpLearnedInfo)


bgpLearnedInfo = ixiangpf.emulation_bgp_info (
    handle                                    = bgpIpv4Peer_2_handle,
    mode                                      = 'learned_info');
pprint(bgpLearnedInfo)
```



Details for BGP Peer 2

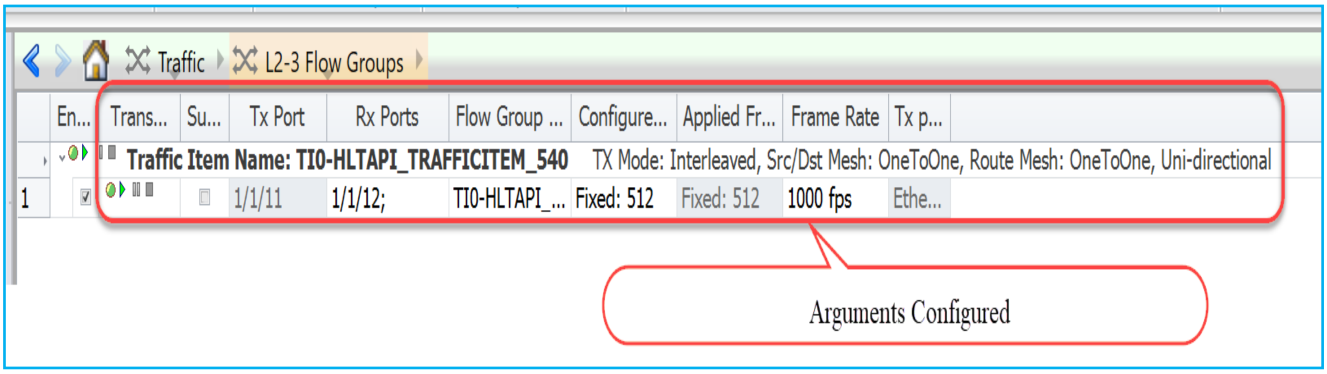
Grouping	Index#	Topology	Device Group	IPv4 Prefix	Prefix Length	Path ID	IPv4 Next Hop	IPv6 Next Hop	MED	Local Preference	Origin	AS Path	Comm
Session 20.	1	BGP_2 Topolo...	BGP_2 Device...										
1/1/12	# 1			200.1.0.0	24	NA	20.20.20.2						

Fig 3.10 Showing details of BGP learned routes

3.11 Configure Traffic:

ixiangpf.traffic_config: Configures the traffic streams on the specified ports with specified options

```
trafficConfig = ixiangpf.traffic_config (
    mode                                      ='create',
    traffic_generator                         ='ixnetwork_540',
    endpointset_count                         =1,
    emulation_src_handle                      =networkGroup_1_handle,
    emulation_dst_handle                      =networkGroup_3_handle,
    track_by                                  ='sourceDestEndpointPair0 trackingenabled0',
    rate_pps                                  =1000,
    frame_size                                =512,
)
```



Traffic Item Name: TI0-HLTAPI_TRAFFICITEM_540 TX Mode: Interleaved, Src/Dst Mesh: OneToOne, Route Mesh: OneToOne, Uni-directional

	En...	Trans...	Su...	Tx Port	Rx Ports	Flow Group ...	Configure...	Applied Fr...	Frame Rate	Tx p...
1	☑		□	1/1/11	1/1/12;	TI0-HLTAPI_...	Fixed: 512	Fixed: 512	1000 fps	Ethe...

Fig 3.11 L2-3 Traffic configured with the specified options

3.12 Start Traffic and Get Statistics:

ixiangpf.traffic_control: Start/stop traffic and allows to modify global traffic options

```
trafficControl = ixiangpf.traffic_control (
    Action                                              ='run',
    traffic_generator                                   ='ixnetwork_540',
    type                                                =['l23']
)
```

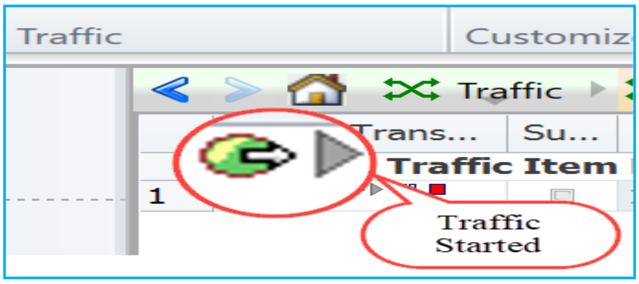


Fig 3.12 Running traffic

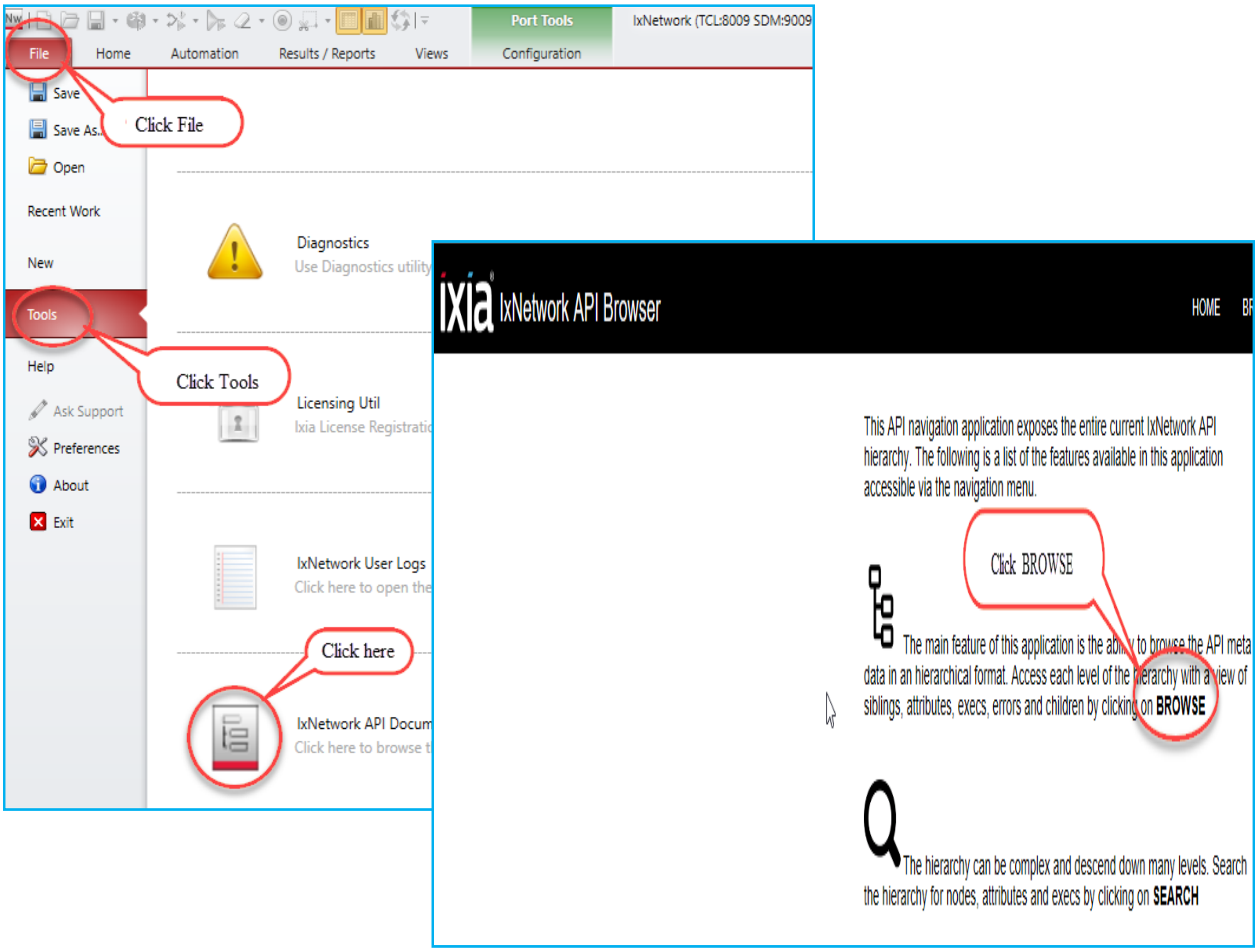
ixiangpf.traffic_stats: Collect Traffic statistics with the specified options

```
protostats = ixiangpf.traffic_stats (
    mode                                                = 'all',
    traffic_generator                                   = 'ixnetwork_540',
    measure_mode                                        = 'mixed'
)
```

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 on clicking on BROWSE.



Click File

Click Tools

Click here

Click BROWSE

This API navigation application exposes the entire current IxNetwork API hierarchy. The following is a list of the features available in this application accessible via the navigation menu.

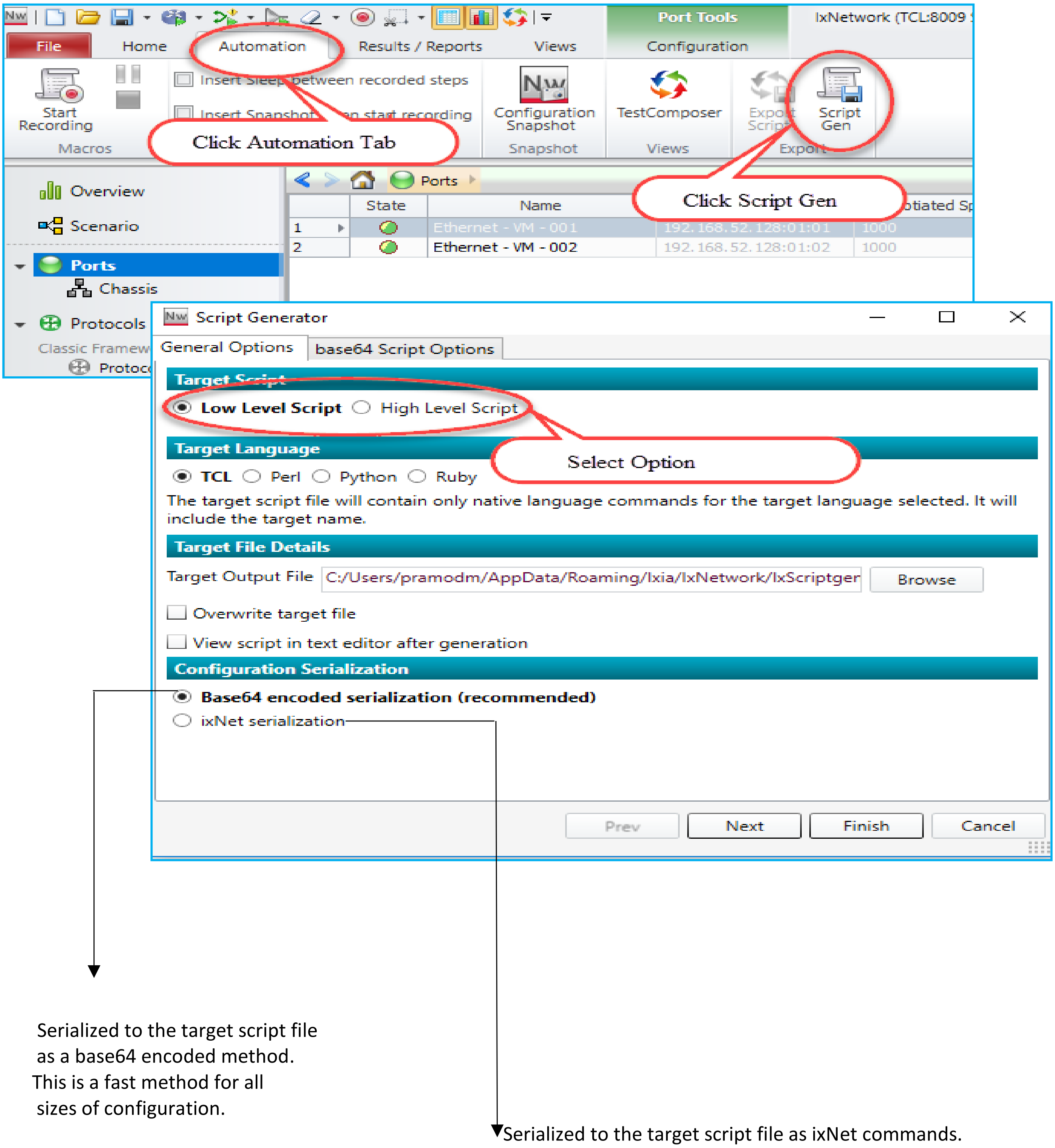
The main feature of this application is the ability to browse the API meta data in an hierarchical format. Access each level of the hierarchy with a view of siblings, attributes, execs, errors and children by clicking on **BROWSE**

The hierarchy can be complex and descend down many levels. Search the hierarchy for nodes, attributes and execs by clicking on **SEARCH**

Fig 4.1 IxNetwork API documentation link

4.2 Script Gen:

- ➢ ScriptGen is a tool that may be used to generate a script that reflects the current configuration of IxNetwork.
- ➢ It is intended to be used after IxNetwork has been successfully configured. The generated scripts can be used to re-create a configuration as the basis for a new test.



Click Automation Tab

Click Script Gen

	State	Name		
1	●	Ethernet - VM - 001	192.168.52.128:01:01	1000
2	●	Ethernet - VM - 002	192.168.52.128:01:02	1000

Script Generator

General Options | base64 Script Options

Target Script

● **Low Level Script** ○ High Level Script

Select Option

Target Language

● **TCL** ○ Perl ○ Python ○ Ruby

The target script file will contain only native language commands for the target language selected. It will include the target name.

Target File Details

Target Output File: C:/Users/pramodm/AppData/Roaming/Ixia/IxNetwork/IxScriptger [Browse]

□ Overwrite target file

□ View script in text editor after generation

Configuration Serialization

● **Base64 encoded serialization (recommended)**

○ ixNet serialization

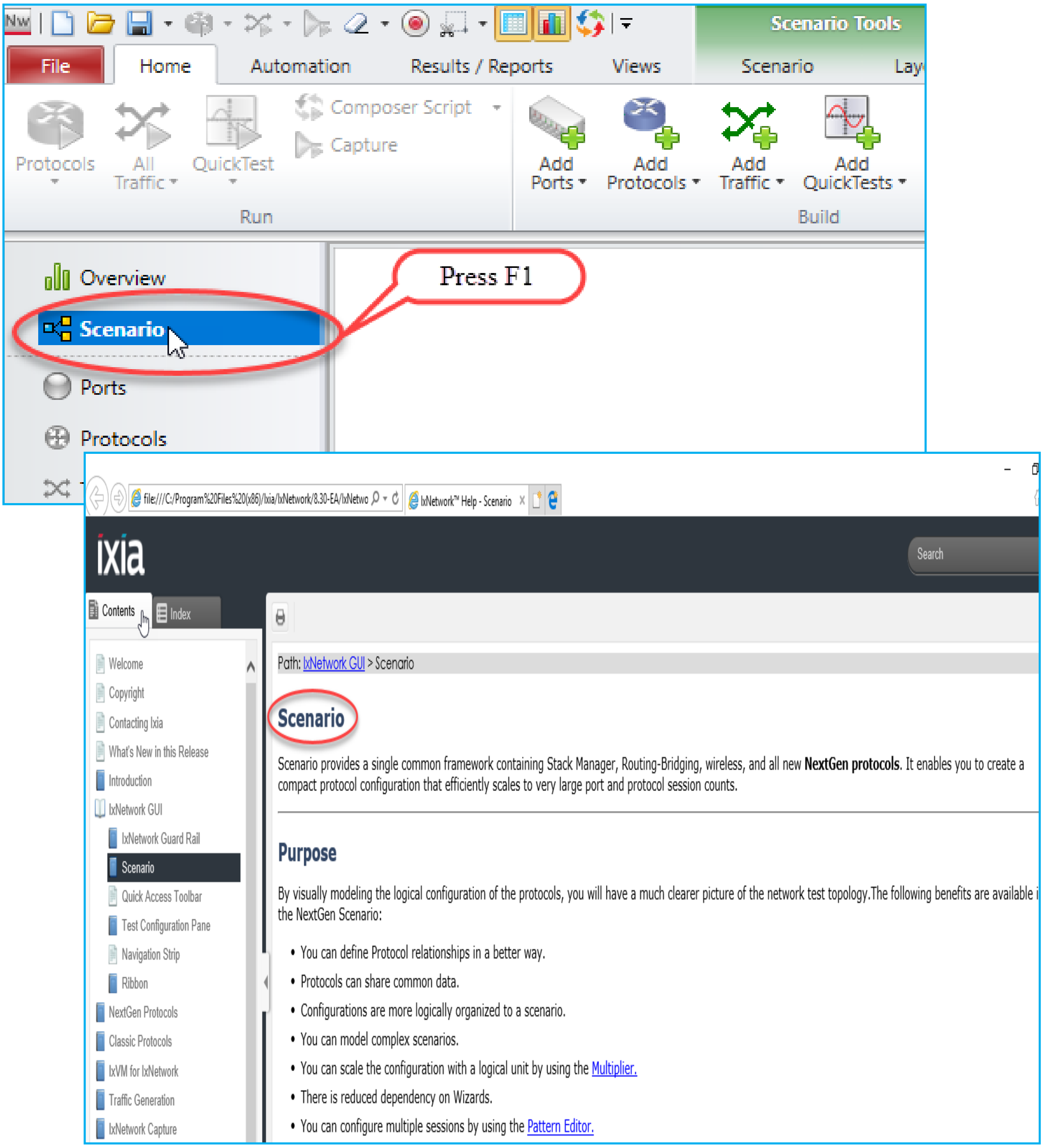
Serialized to the target script file as a base64 encoded method. This is a fast method for all sizes of configuration.

Serialized to the target script file as ixNet commands.

Fig 4.2 IxNetwork ScriptGen link

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.



Path: IxNetwork GUI > Scenario

Scenario

Scenario provides a single common framework containing Stack Manager, Routing-Bridging, wireless, and all new **NextGen protocols**. It enables you to create a compact protocol configuration that efficiently scales to very large port and protocol session counts.

Purpose

By visually modeling the logical configuration of the protocols, you will have a much clearer picture of the network test topology.The following benefits are available i the NextGen Scenario:

- You can define Protocol relationships in a better way.
- Protocols can share common data.
- Configurations are more logically organized to a scenario.
- You can model complex scenarios.
- You can scale the configuration with a logical unit by using the Multiplier.
- There is reduced dependency on Wizards.
- You can configure multiple sessions by using the Pattern Editor.

Fig 4.3 IxNetwork F1 option usability

5. To Know More on NGPF:

https://www.youtube.com/watch?v=A0mbZuP94jo
http://openixia.com/sampleScripts//IxNetwork/HighLevelApi/Ngpf/Python

6. Support:

For more information: https://support.ixiacom.com/
For support assistance, contact : support.ix@keysight.com