IXNETWORK-NGPF
QUICK REFERENCE GUIDE

ixia | A Keysight Business

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

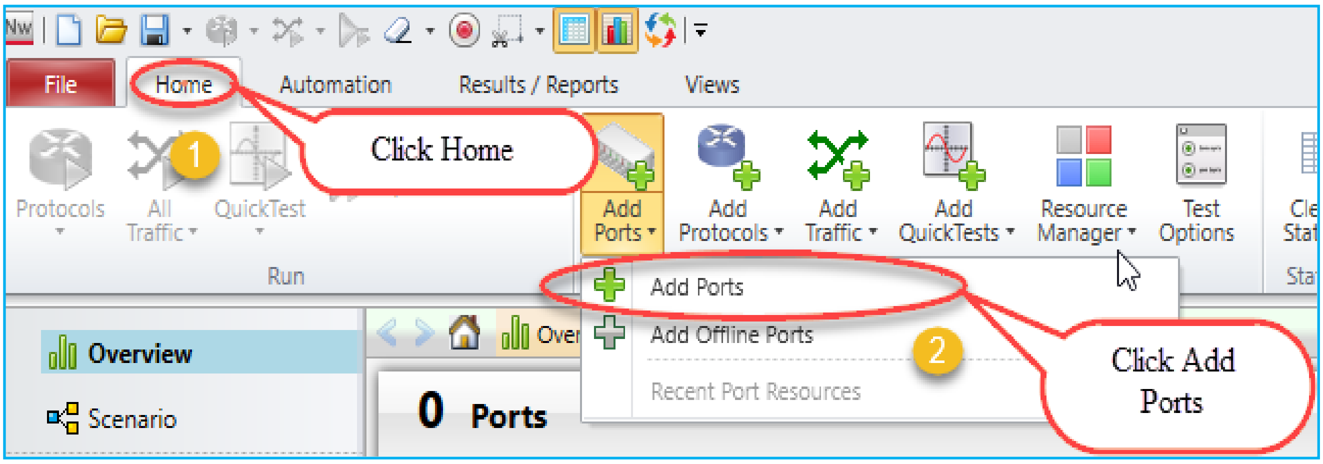


Fig 2.1 Selecting Ports

➢ 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.

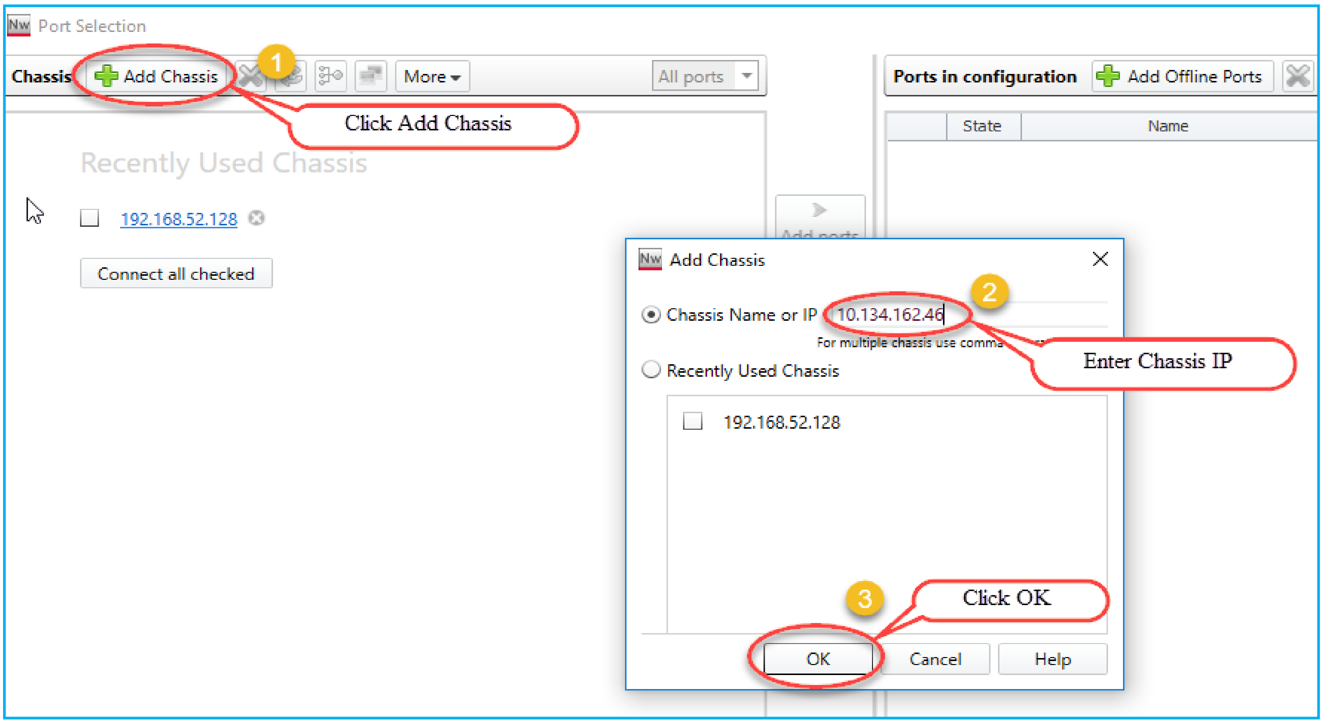


Fig 2.1.1 Selects chassis by entering chassis IP

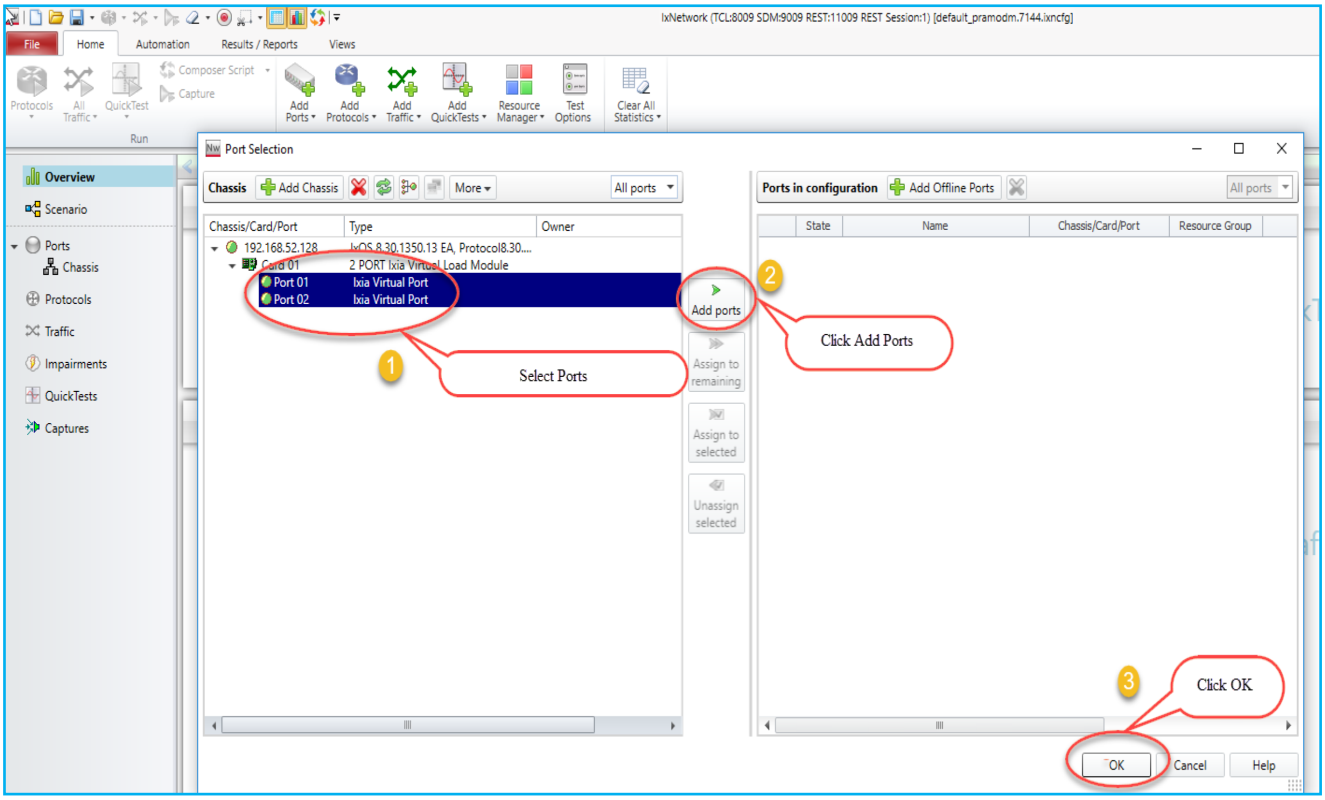
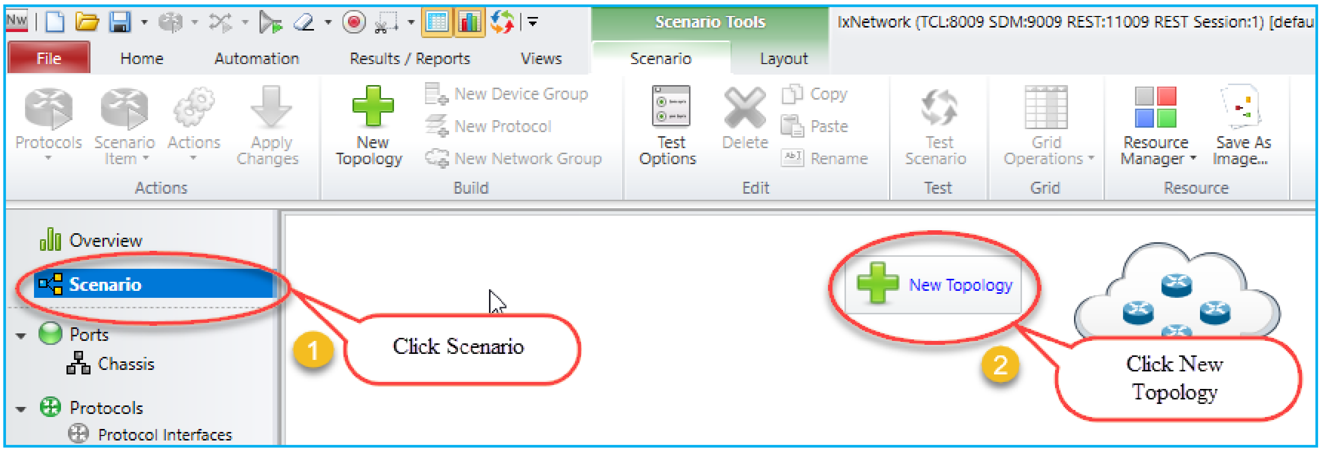


Fig 2.1.2 Port selection

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.



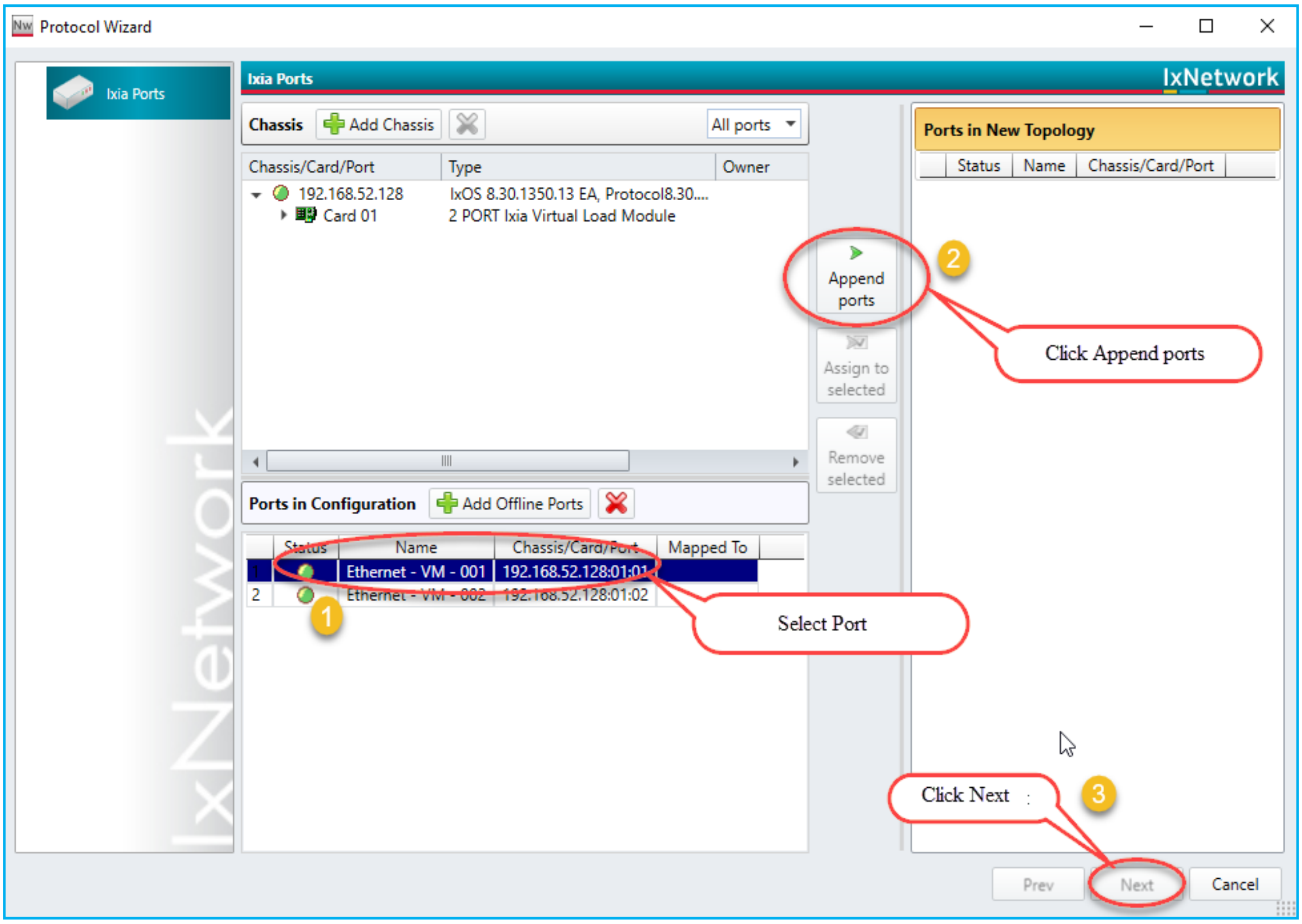


Fig 2.2 Topology with selected ports

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.

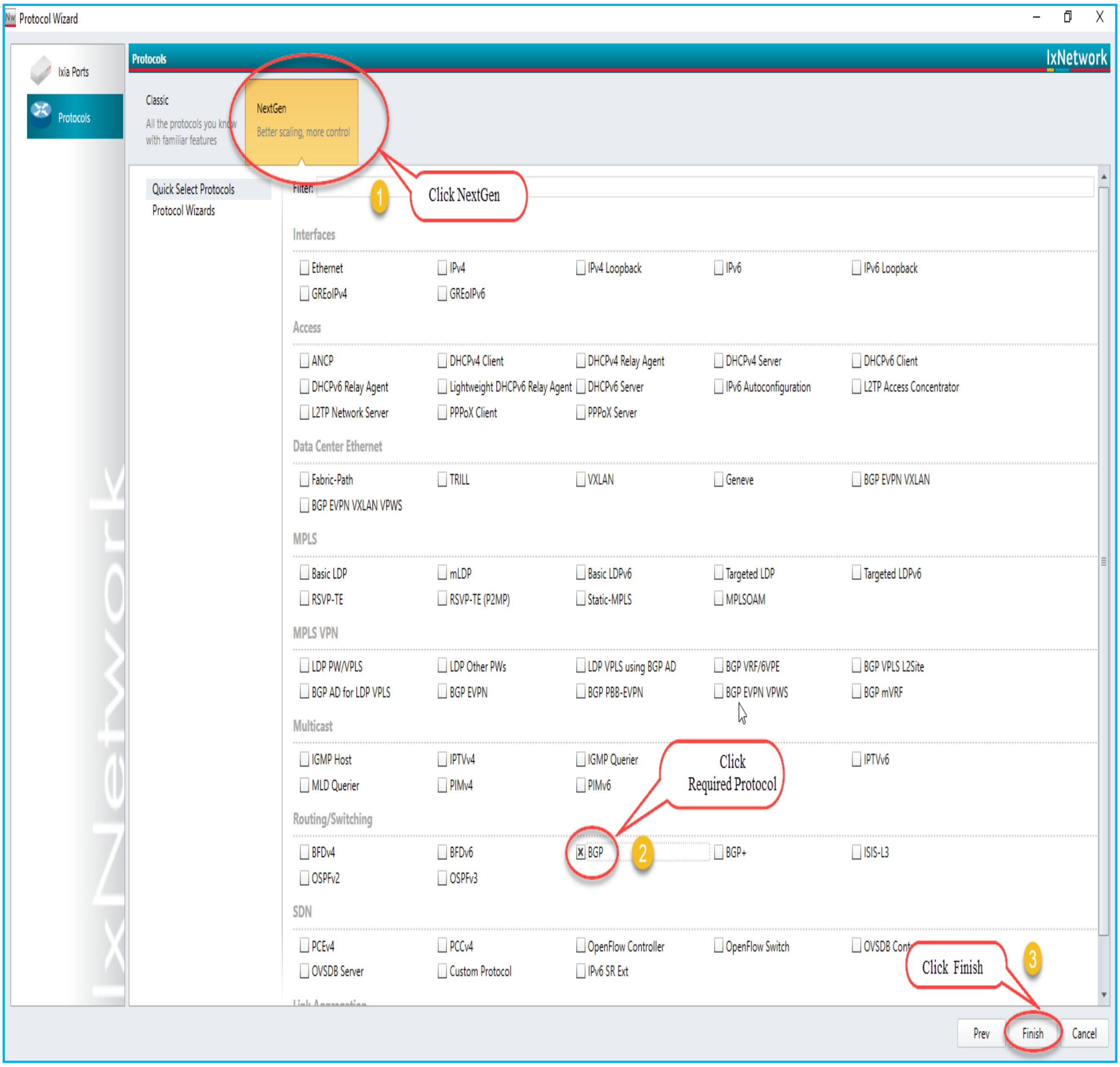


Fig 2.3 Selected BGP protocol

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.

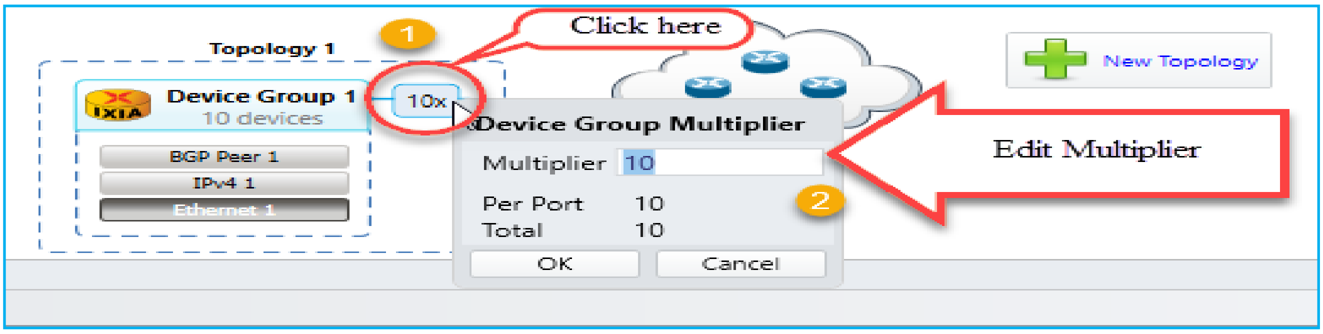


Fig 2.4 Emulate number of devices by using device group multiplier

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.

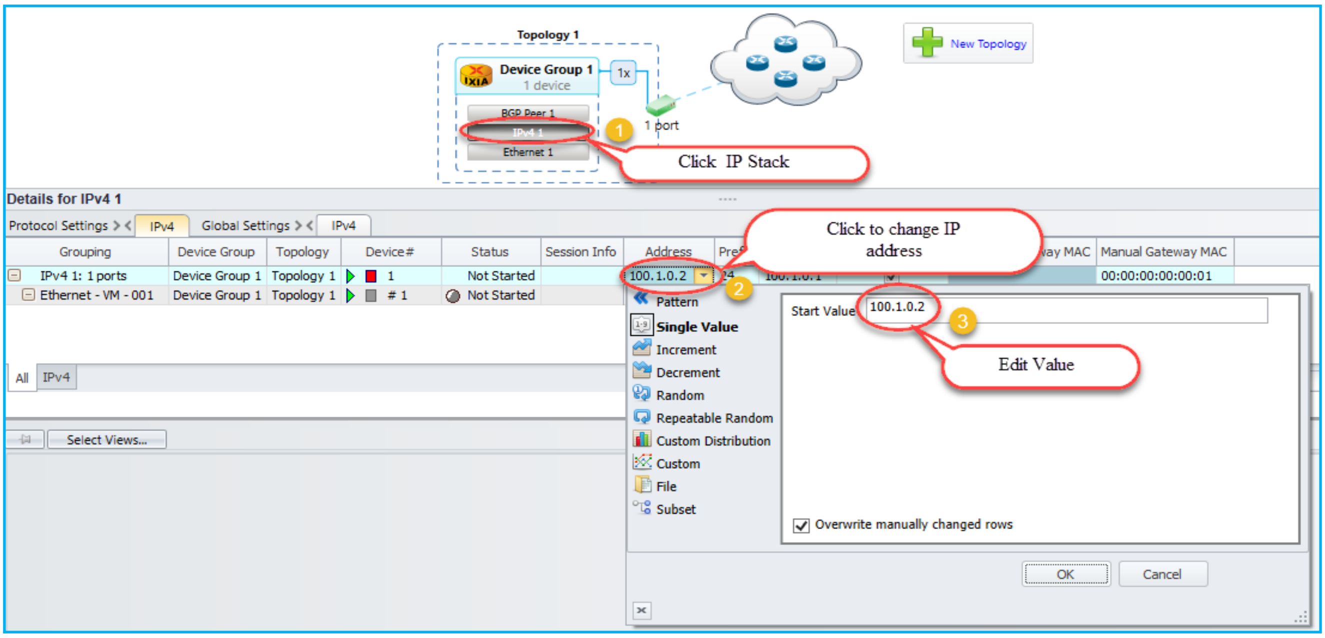


Fig 2.5 Configuring interface sections

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.

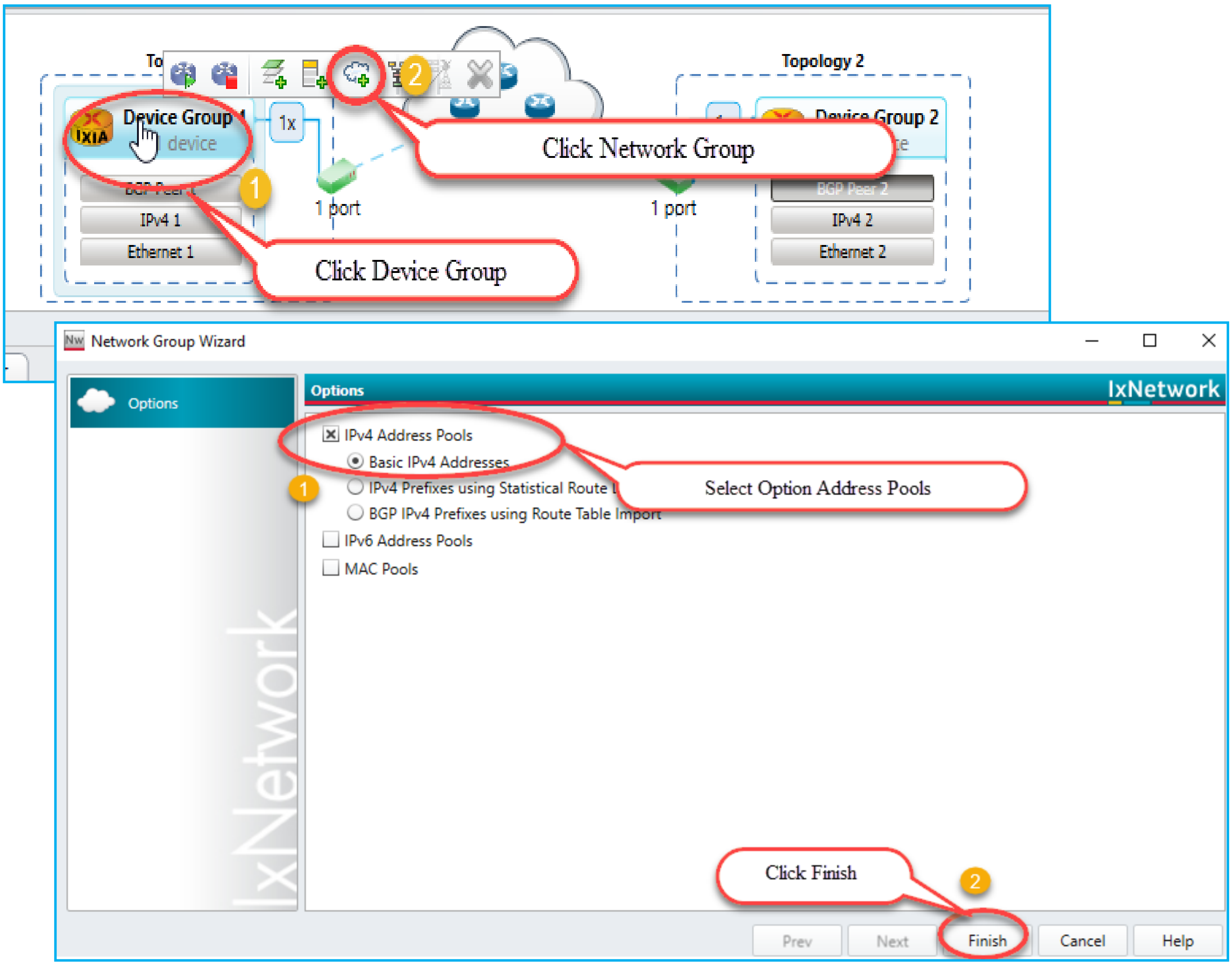


Fig 2.7 Route Profile addition by using network groups

2.8 Start Protocols:

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

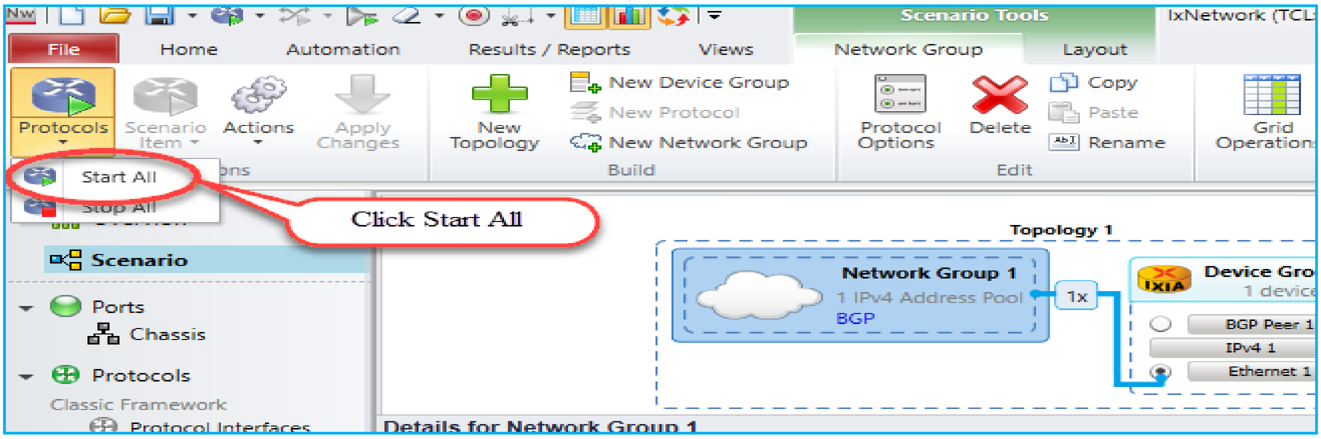


Fig 2.8 Brings up all protocol stacks

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.

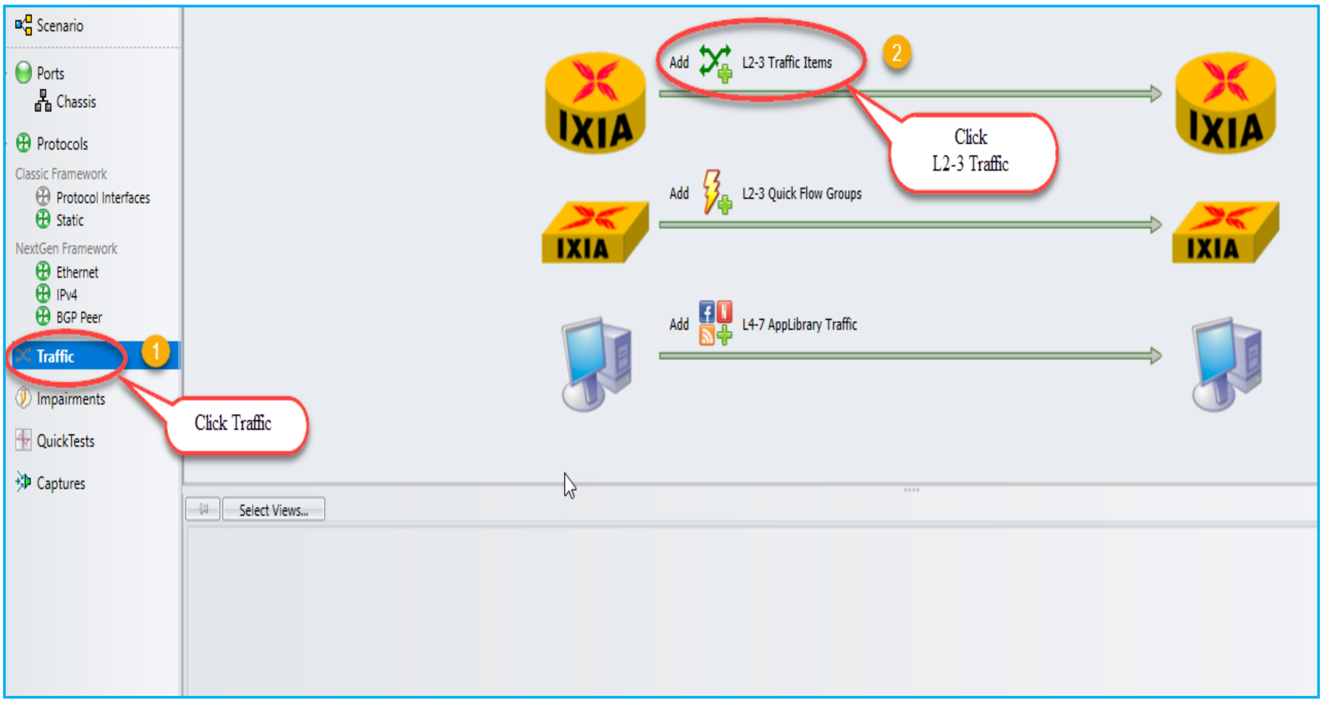


Fig 2.9 Configures L2-3 traffic

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.

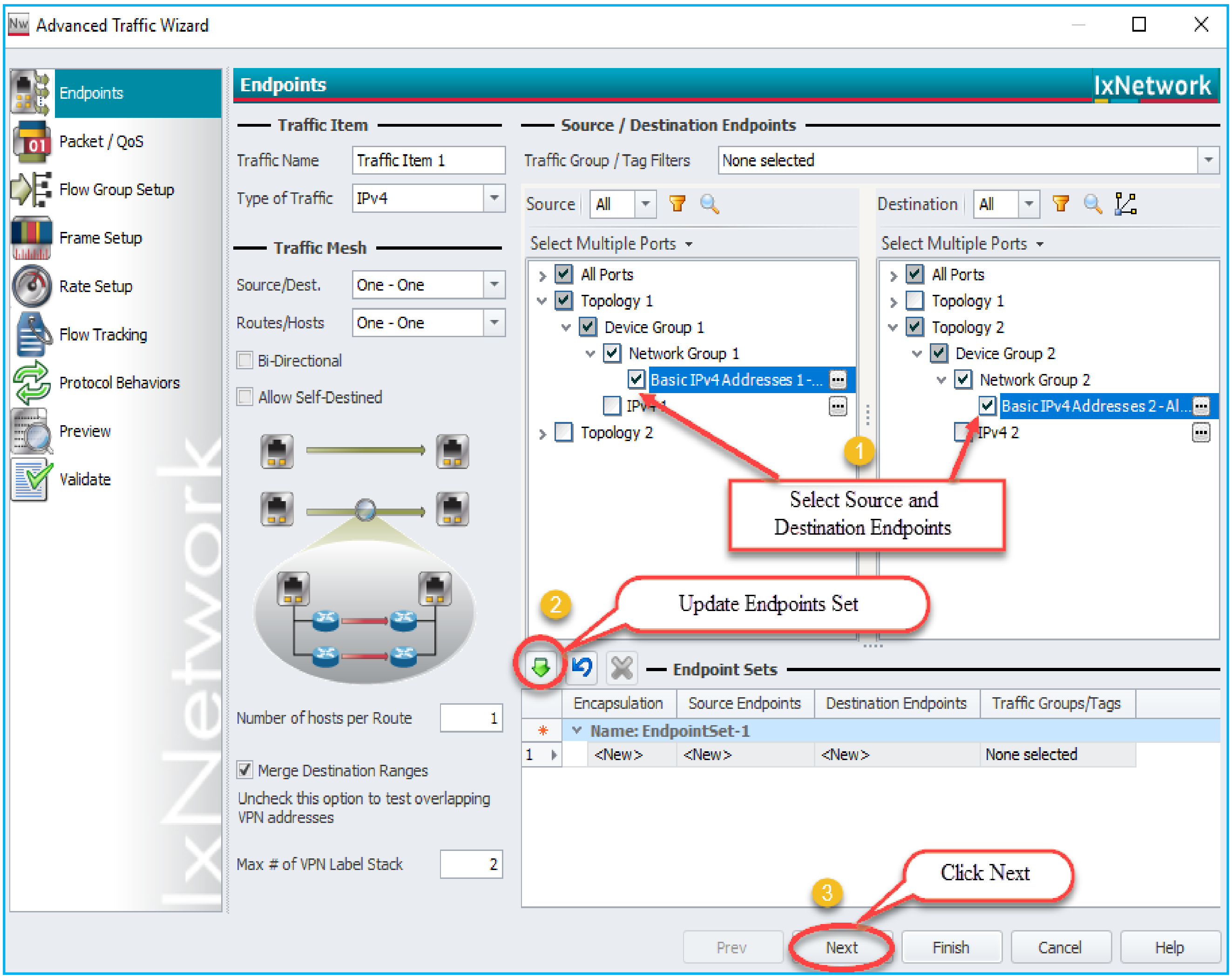
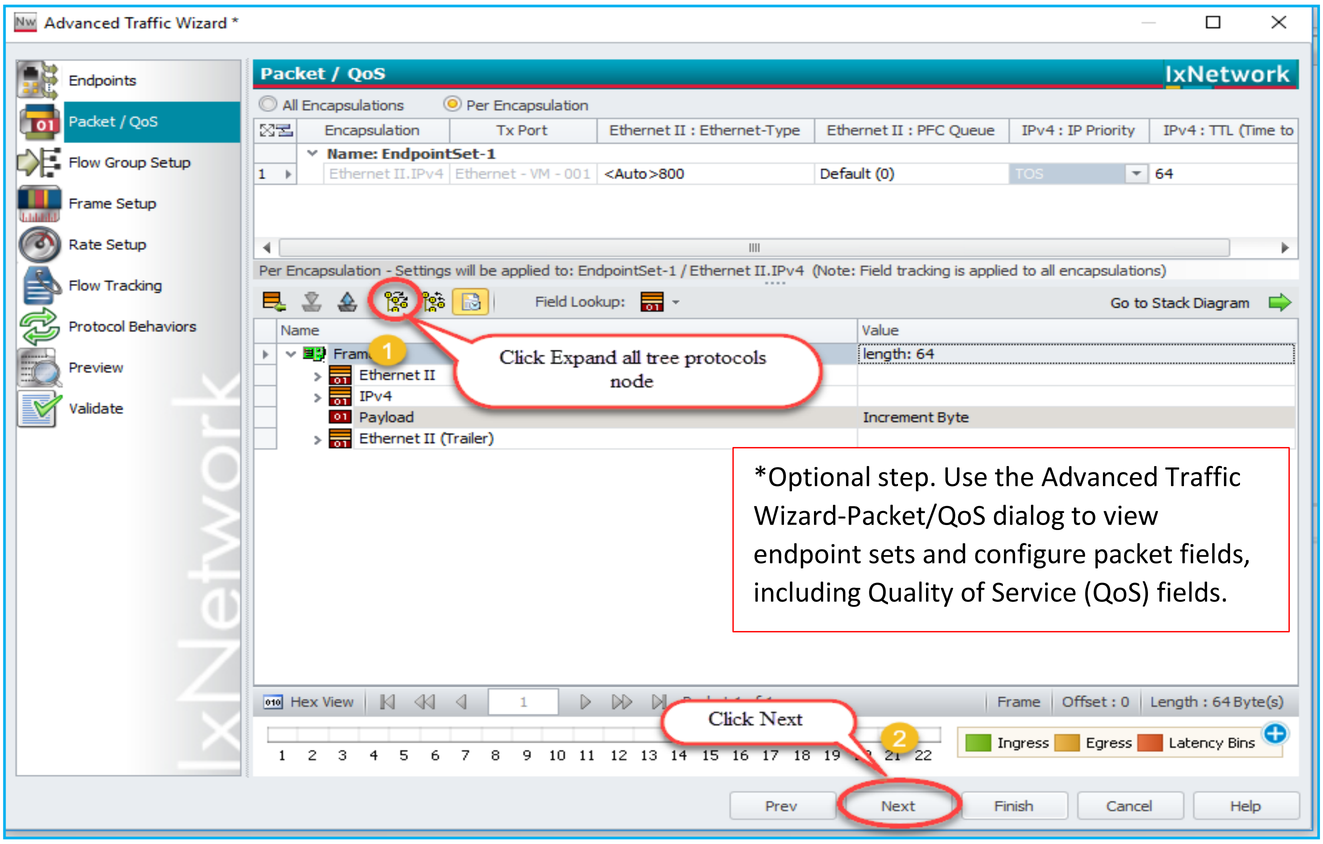


Fig 2.10 Configures source and destinations endpoints set

2.11 Edit Packet and Setup Flow Groups:

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



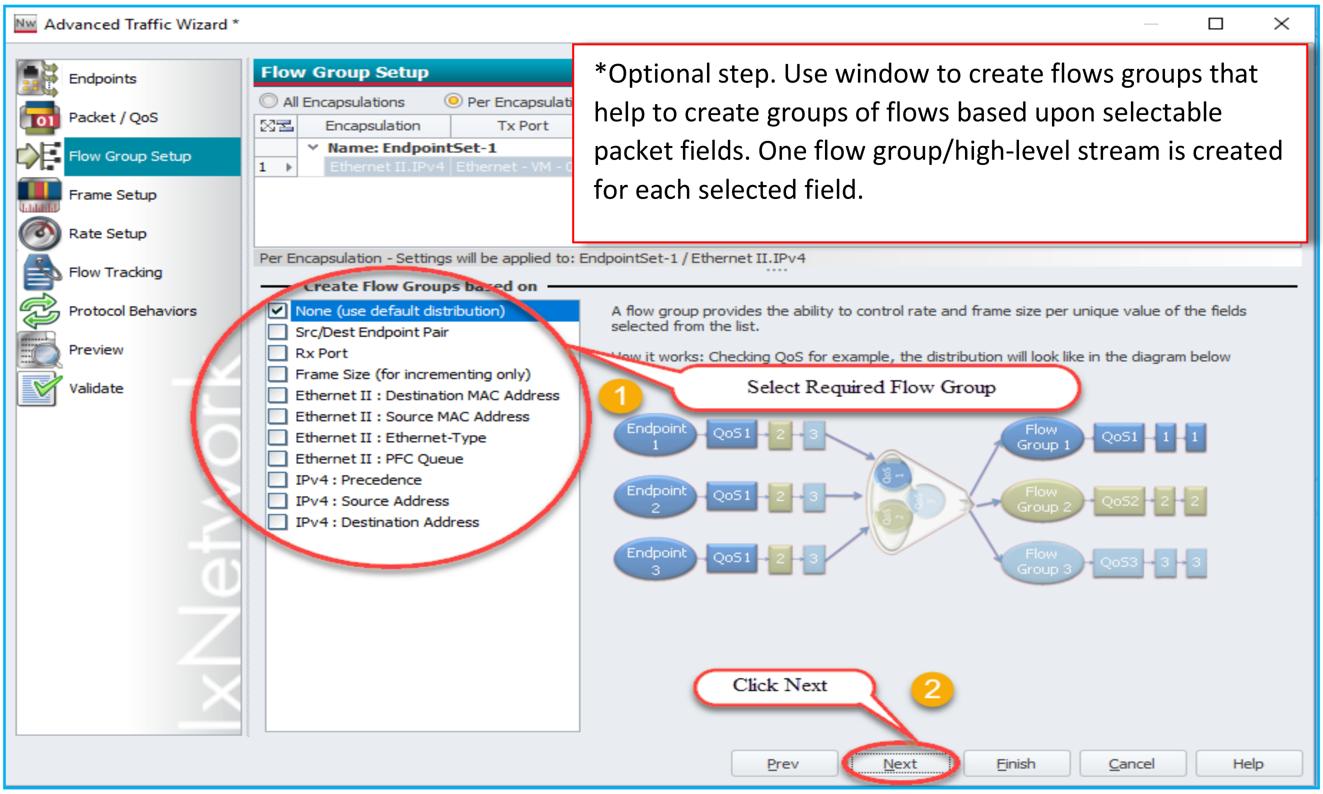
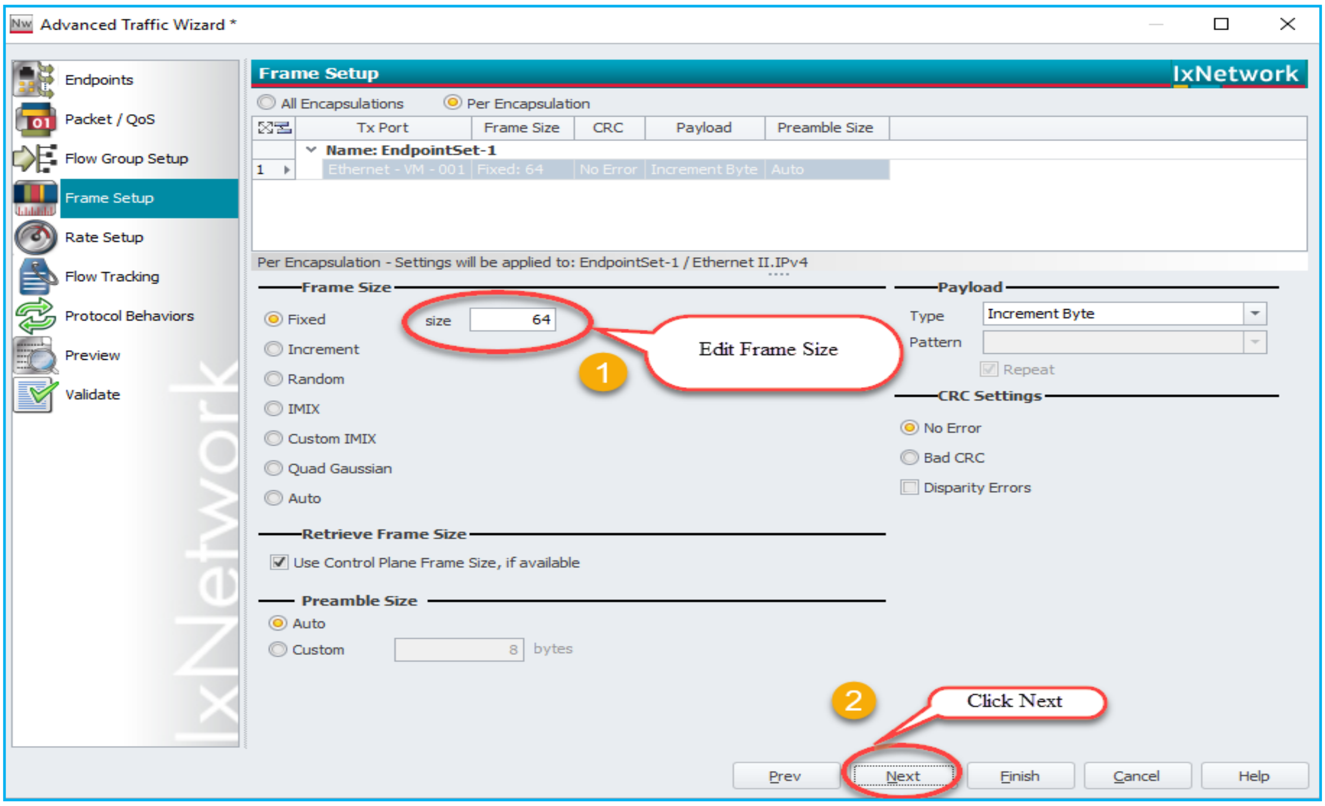


Fig 2.11 Customizing the packet and creating flow groups

2.12 Setup Frame Size and Rate:

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



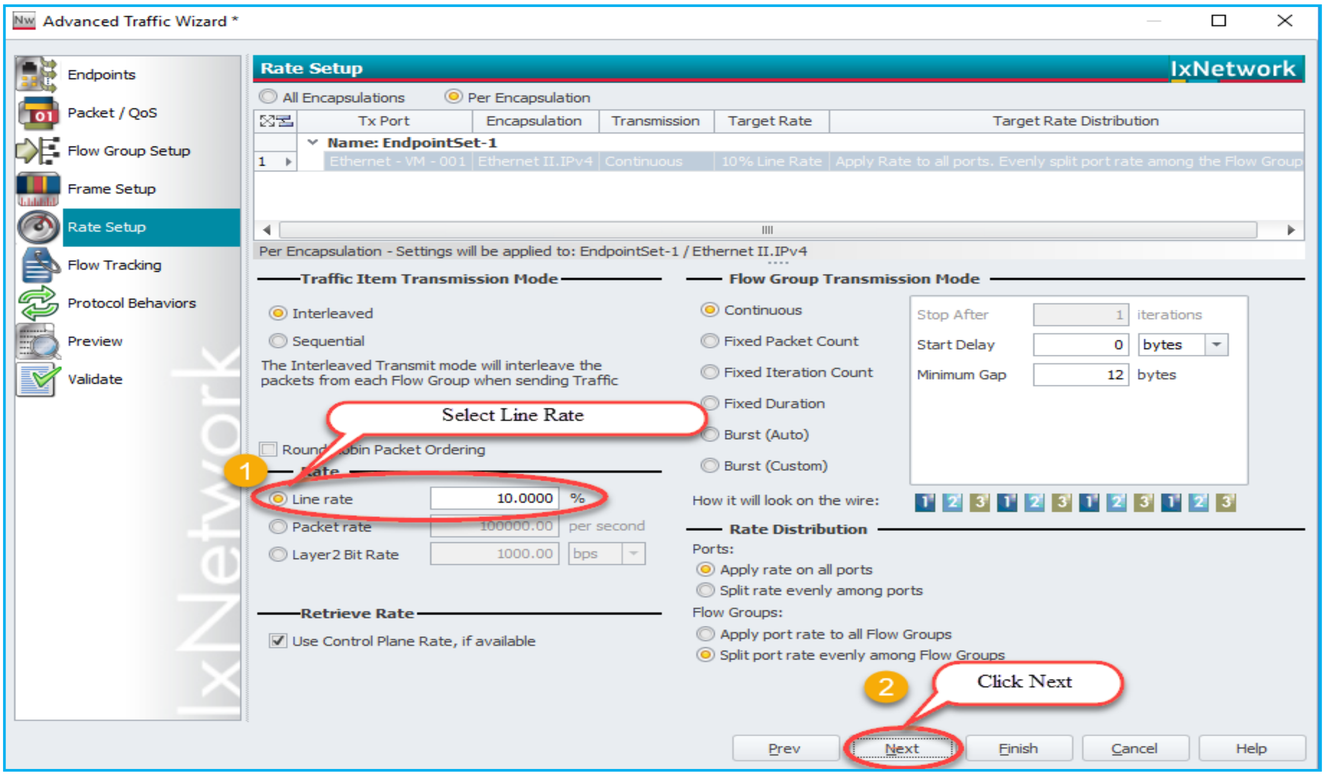
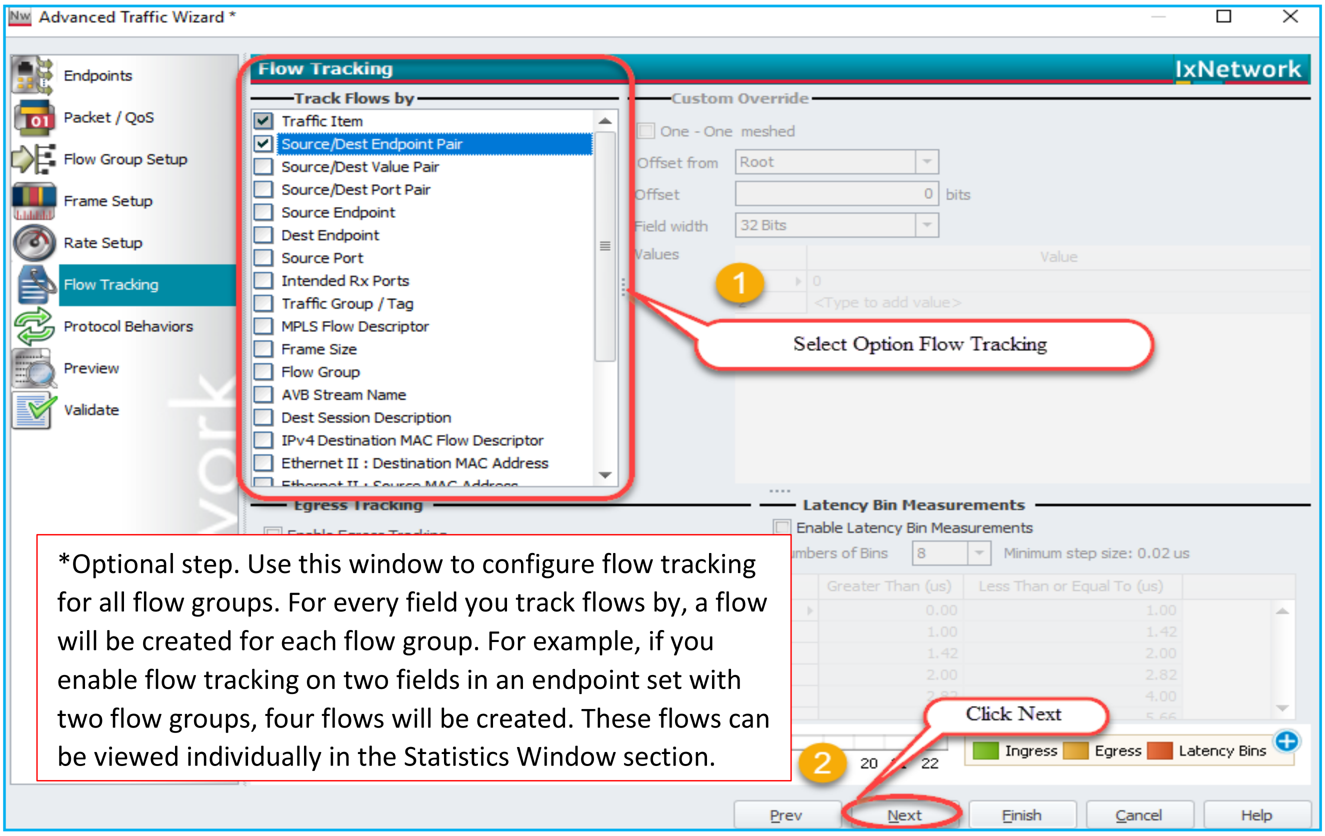


Fig 2.12 Setting up the frame size and line rate of the traffic

2.13 Setup Flow Tracking and Protocol Behavior:

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



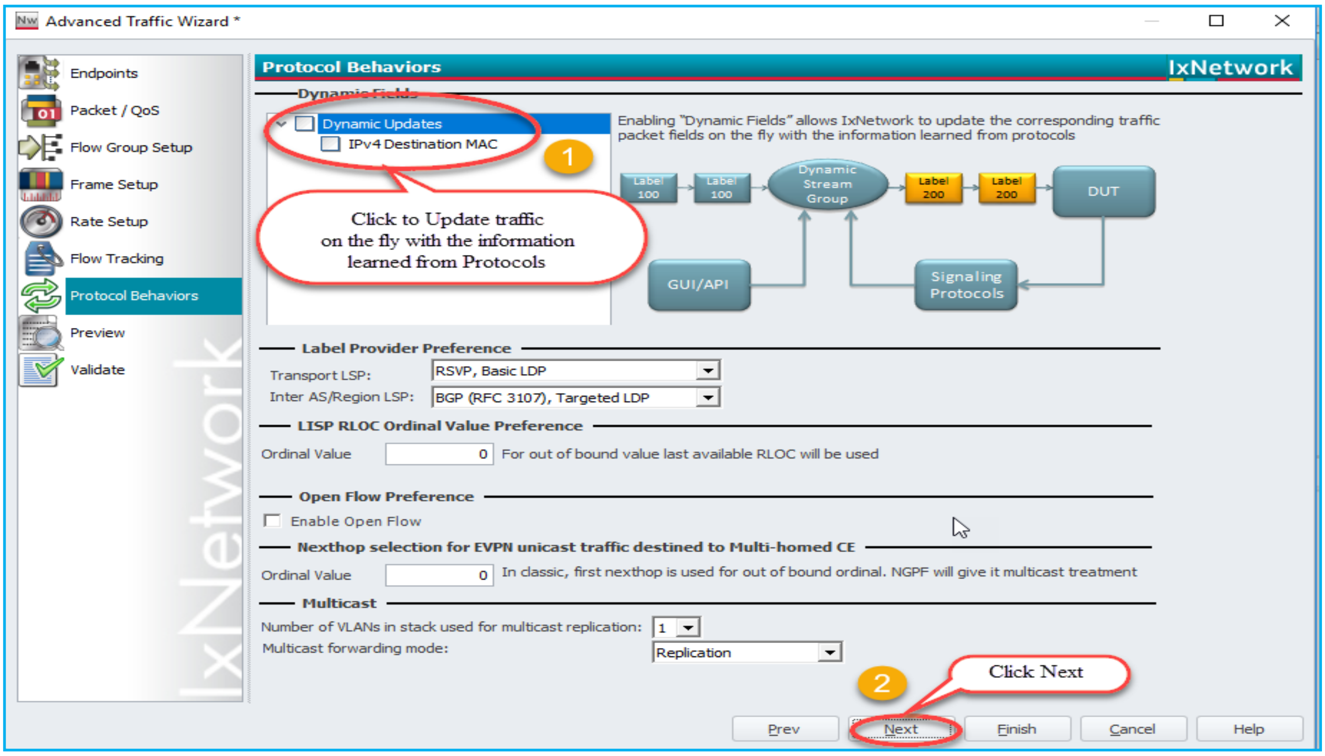
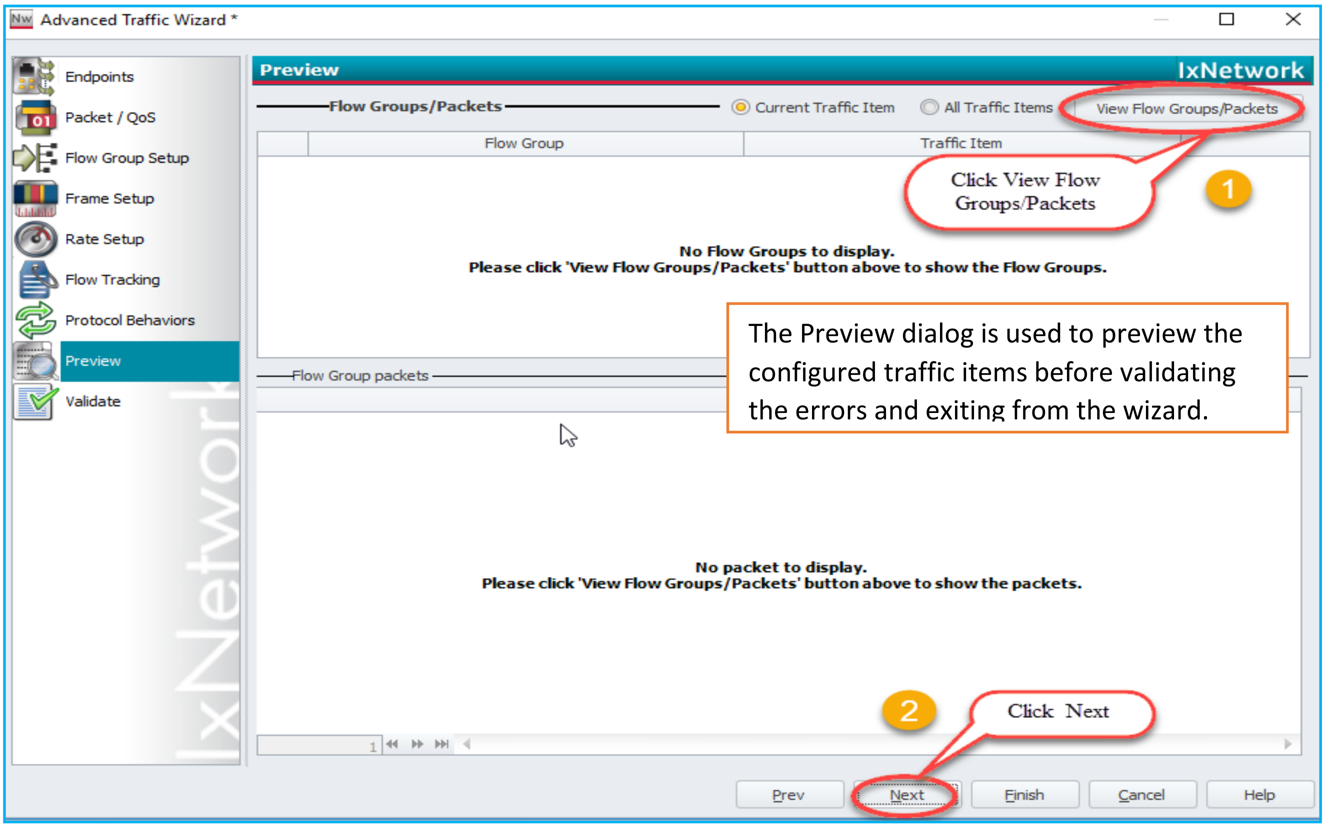


Fig 2.13 Setting up the flow tracks and traffic update option

2.14 Validate Traffic:



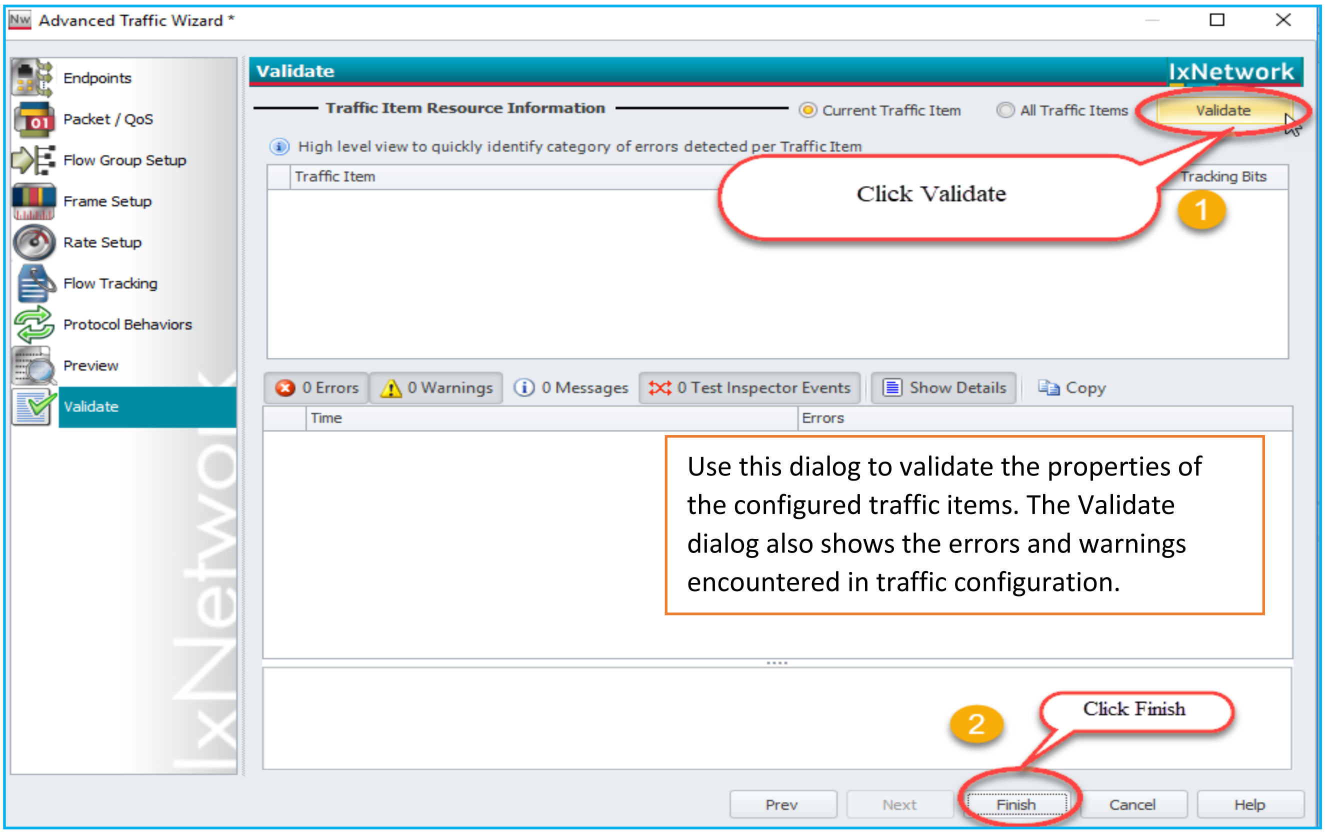


Fig 2.14 Viewing the flow groups and validating the traffic

2.15 Apply Traffic, Start Traffic, and Statistics View:

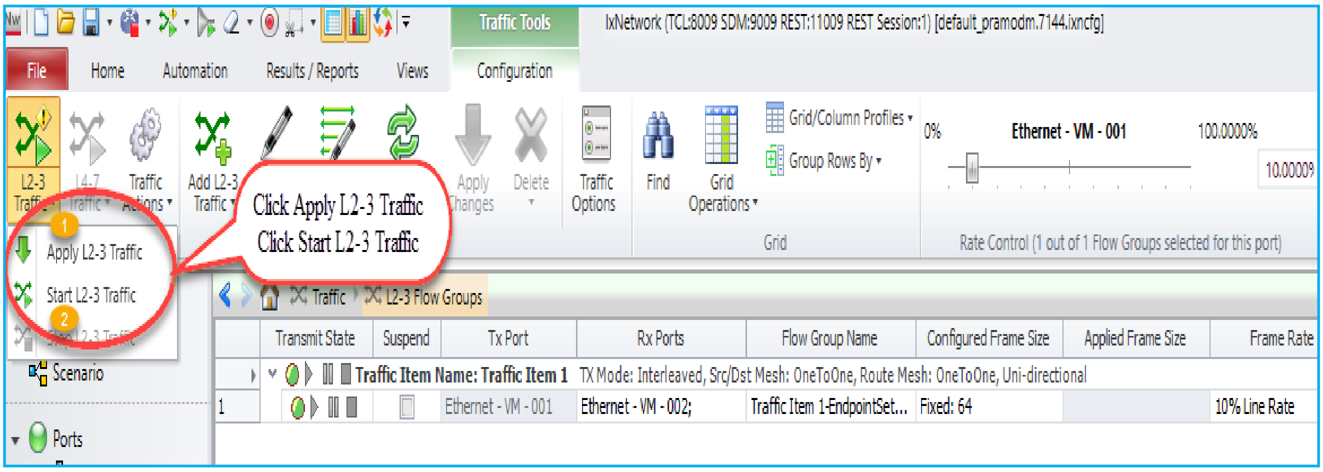


Fig 2.15 Applying and Starting traffic

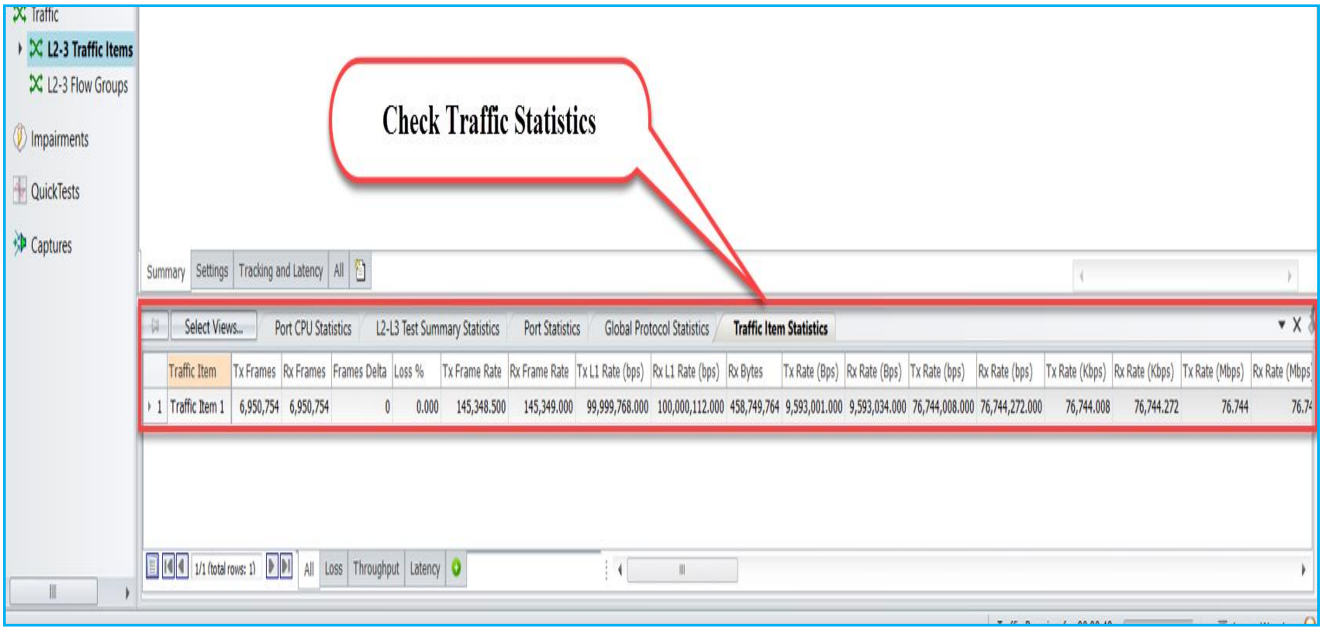


Fig 2.15.1 Check for traffic statistics

3. Configure BGP through Automation (HLPyAPI):

IxNetwork provides a wide array of automation APIs. The REST API is the recommended method for creating new test scripts for IxNetwork. Please refer to the IxNetwork REST API quick reference guide for more details. The legacy API library continue to be supported for existing test scripts. 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                      = "192.168.52.198"
tcl_server                      = "10.38.66.146"
ixnetwork_tcl_server            = "10.38.66.146:8009"
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.*

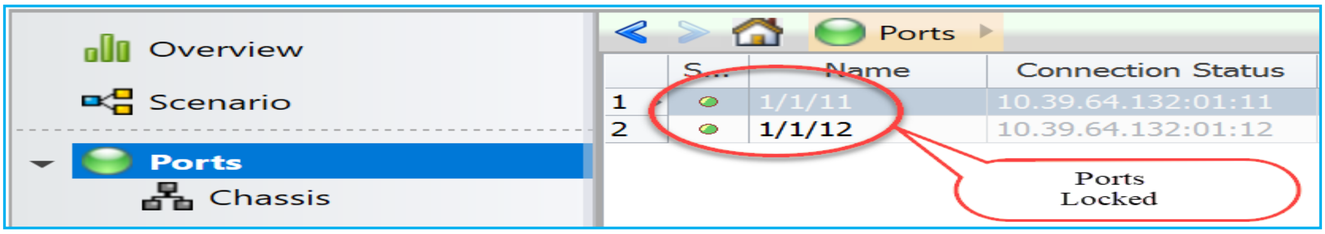


Fig 3.2: Chassis connected and selected ports locked

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']
```

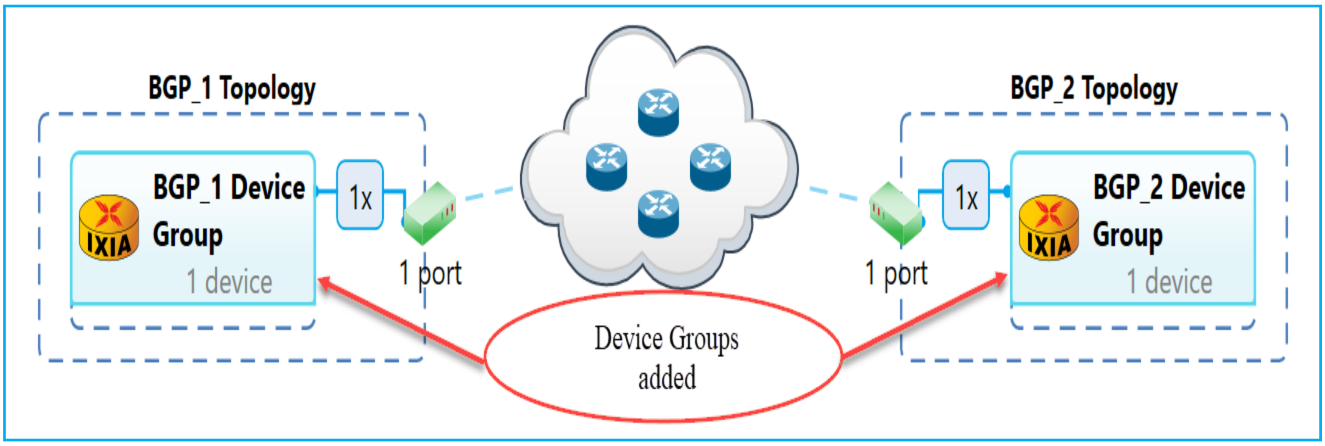


Fig 3.3: Device groups added to respective topologies

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']
```

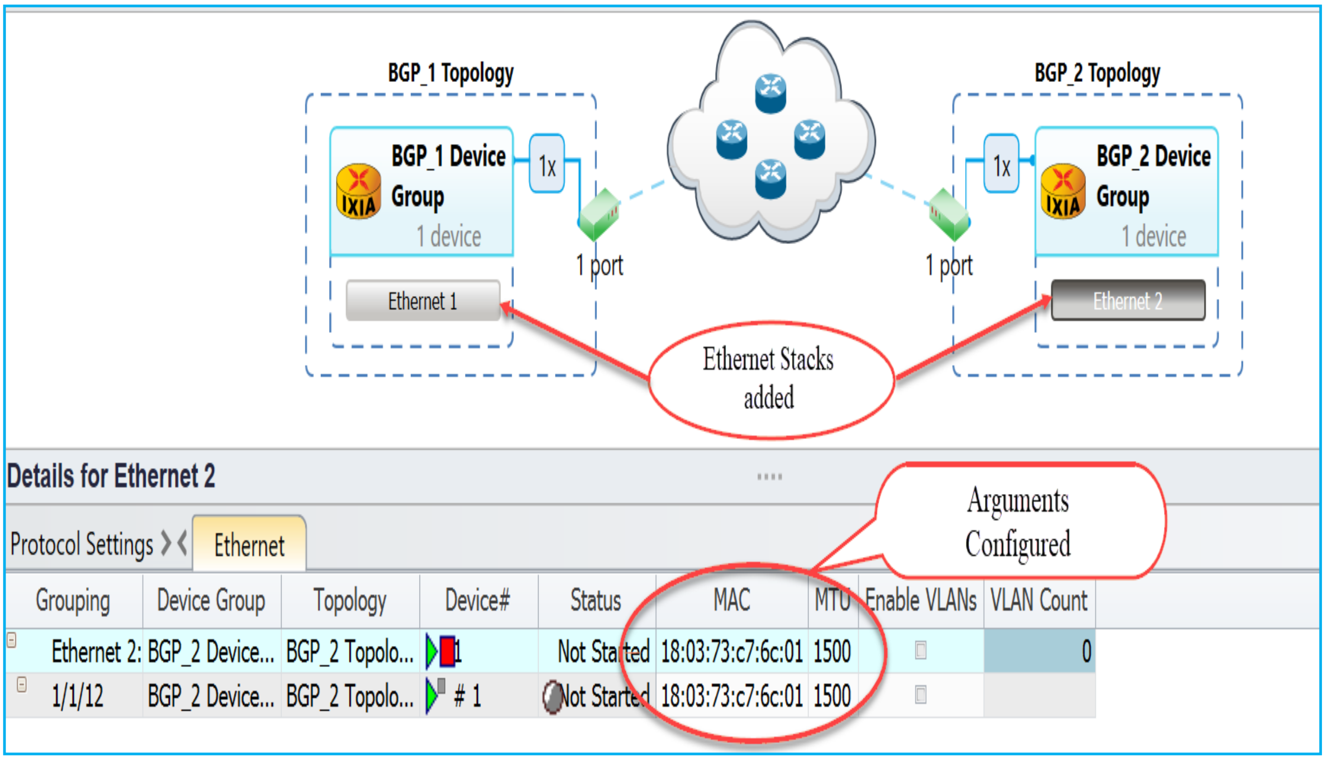


Fig 3.4: Ethernet stacks added to device groups

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']
```

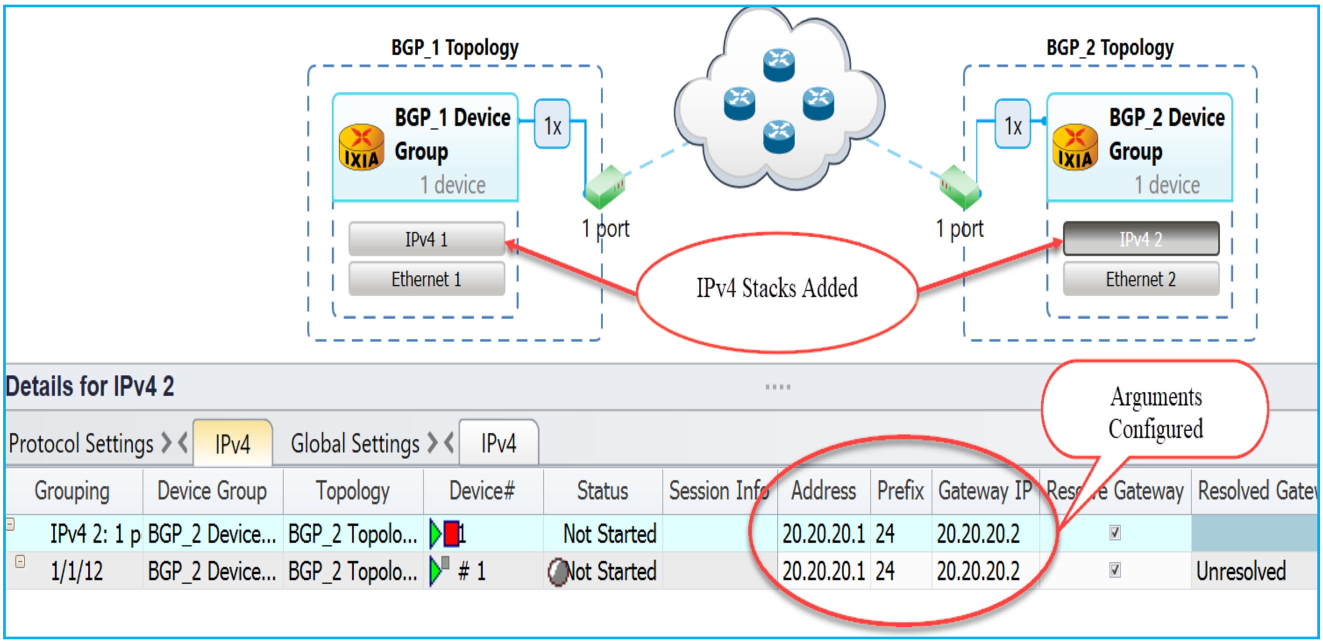


Fig 3.5: IPv4 stacks added to ethernet stacks

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']
```

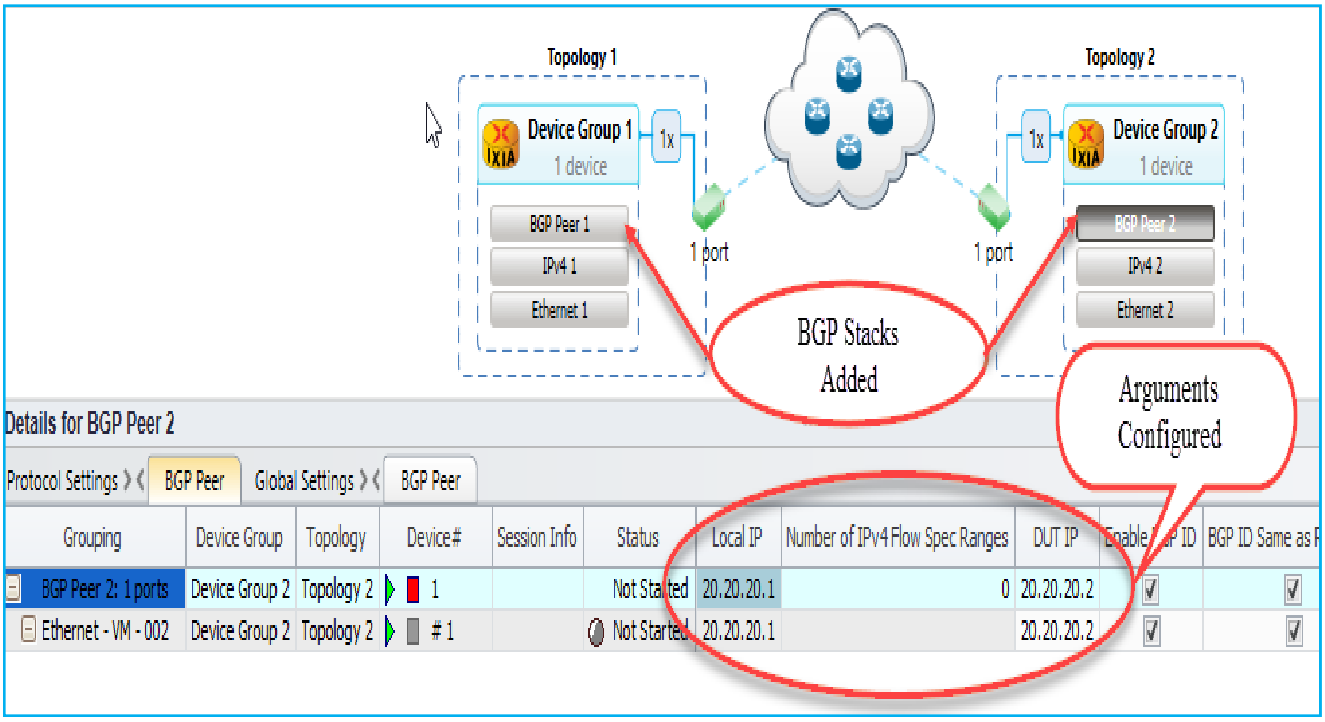


Fig 3.6: BGP stacks added to ipv4 stacks

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’]
```

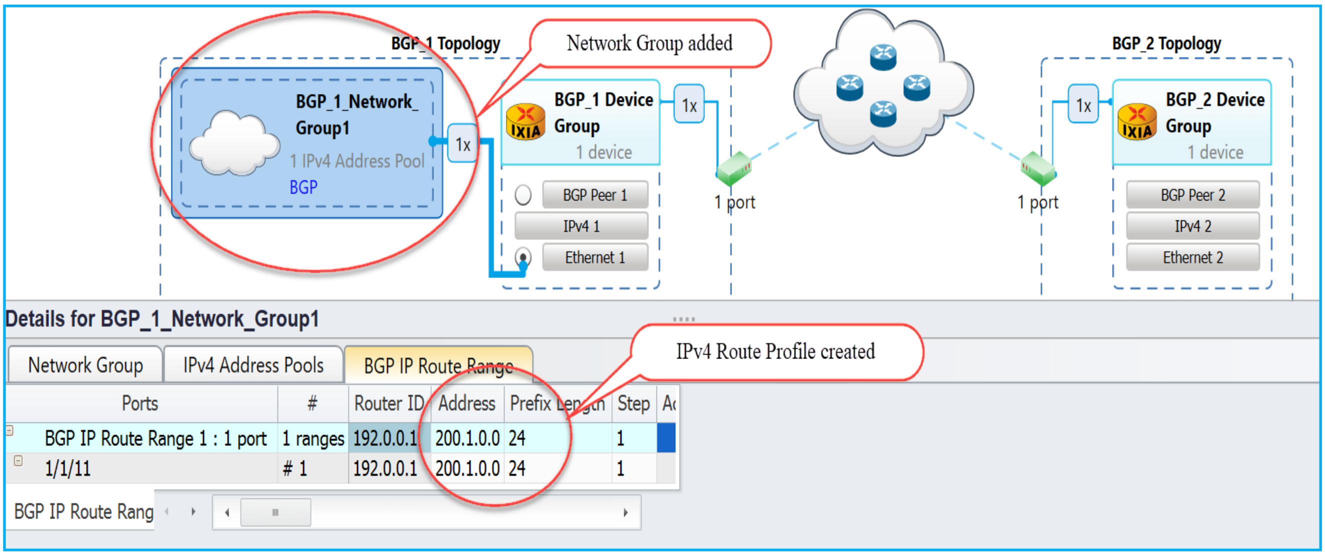


Fig 3.7 Adding BGP network group to device group 1

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']
```

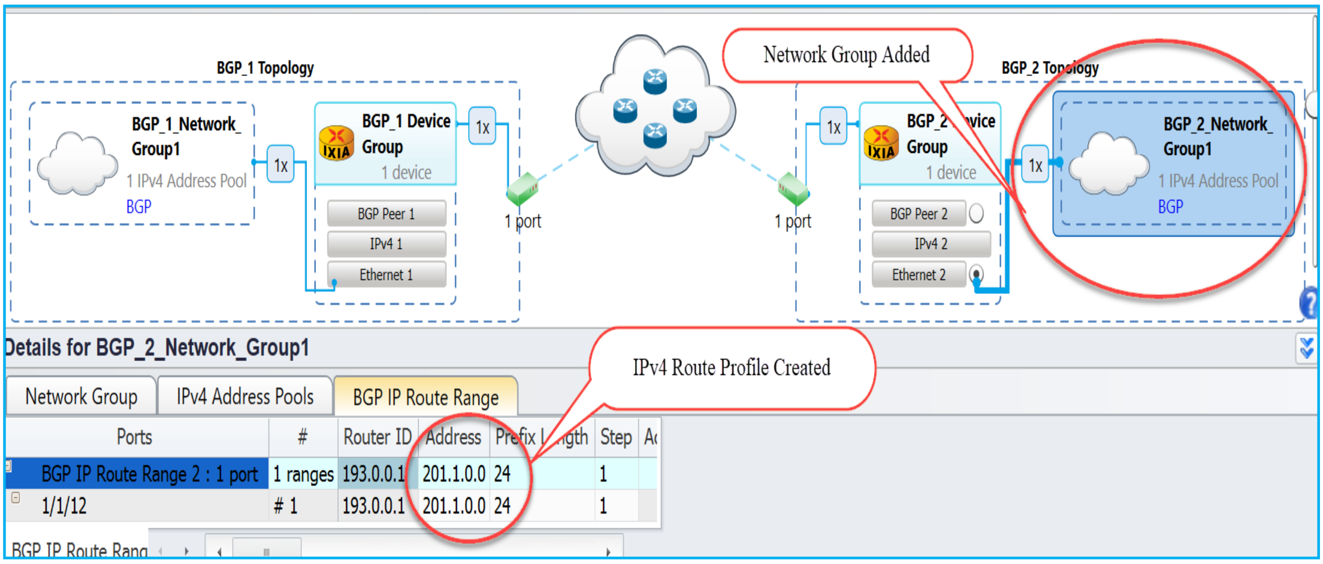


Fig 3.7.1 Adding BGP network group to device group 2

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)
```

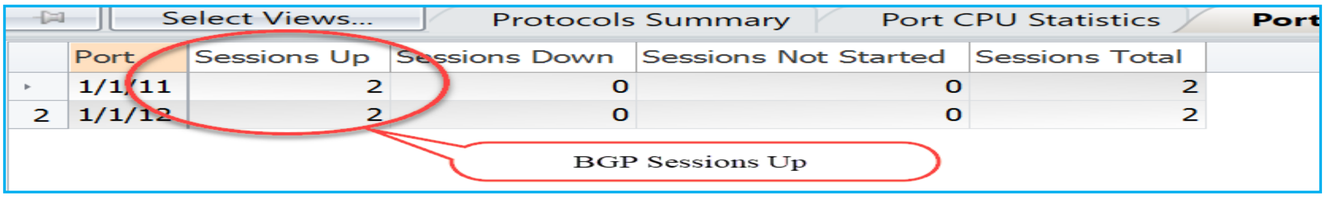


Fig 3.8 Starting all protocol stacks to come up

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',
)
```

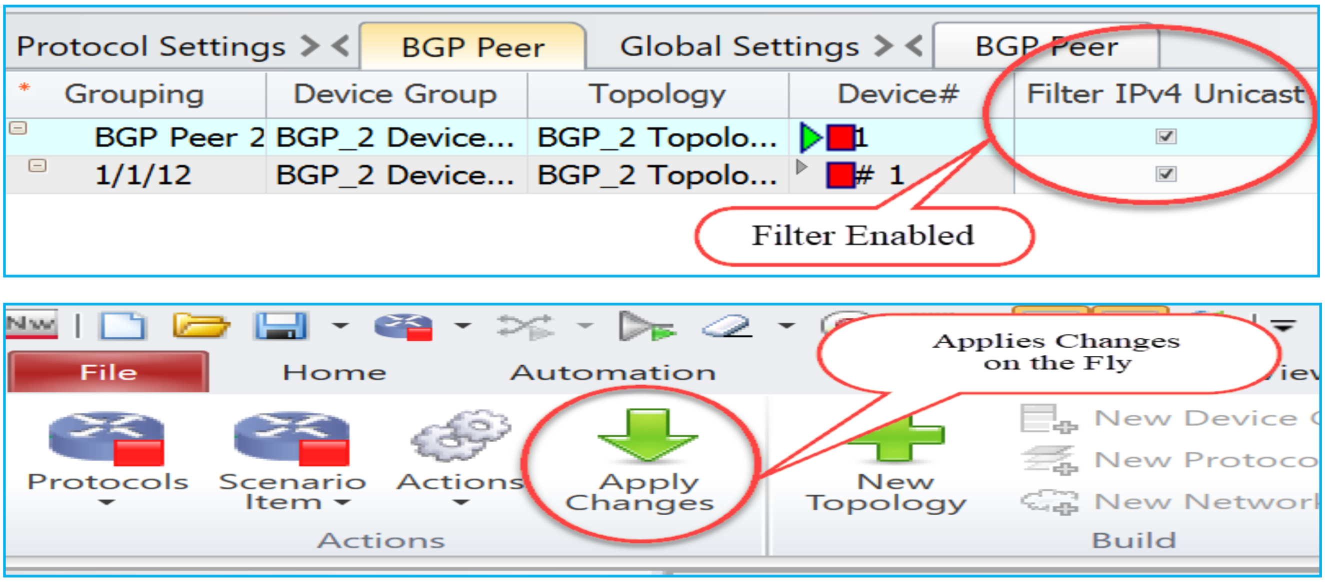


Fig 3.9 Enabling the route filter and applying the changes on the fly

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)
```

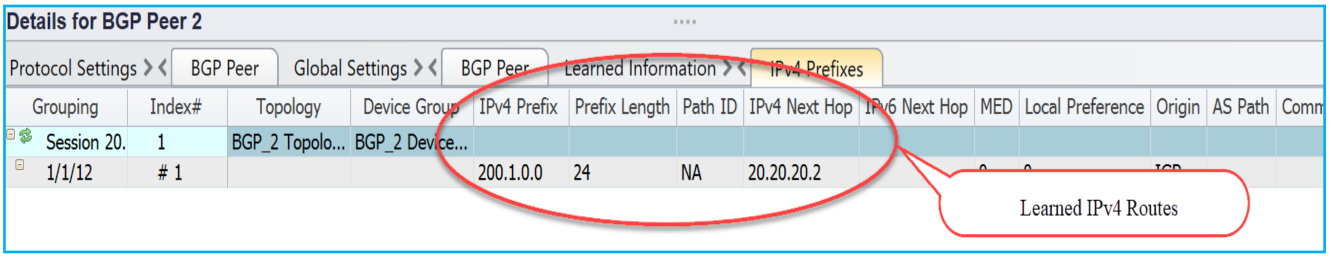


Fig 3.10 Showing details of BGP learned routes

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,
)
```

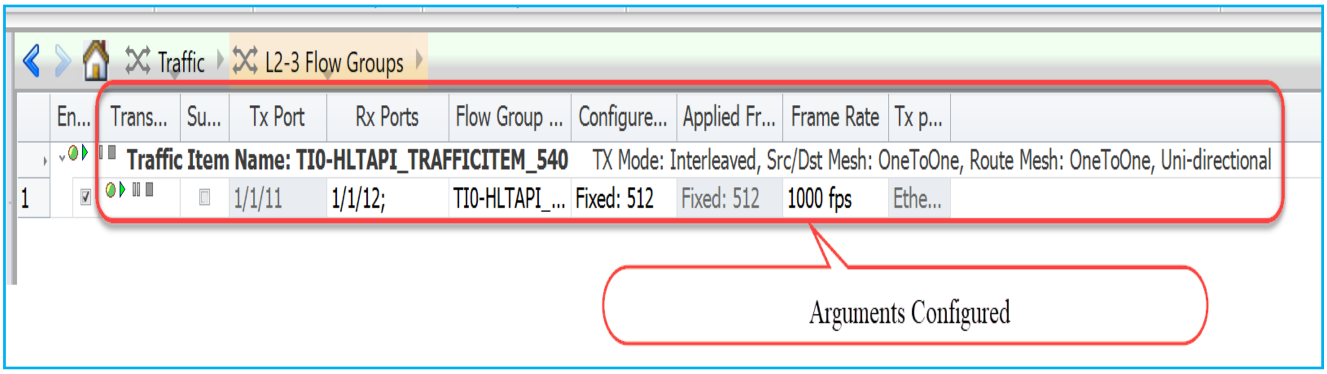


Fig 3.11 L2-3 Traffic configured with the specified options

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']
)
```

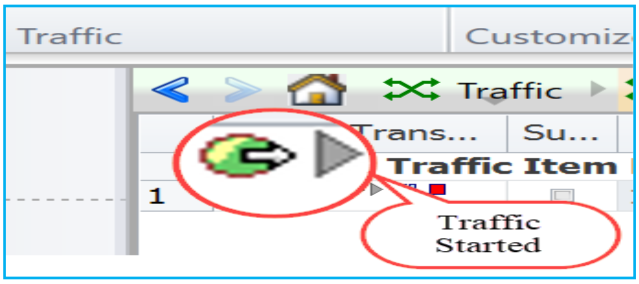


Fig 3.12 Running traffic

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.

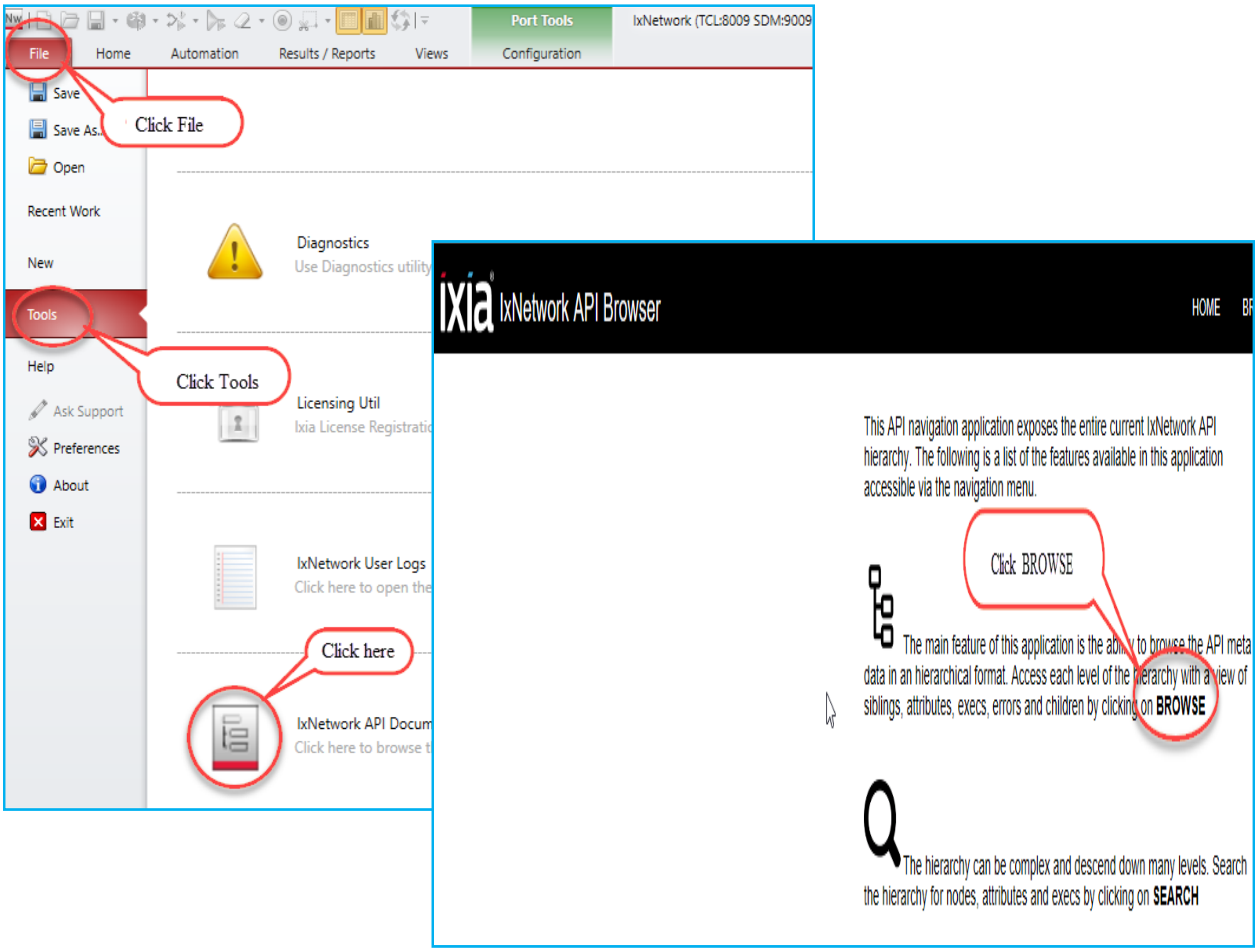


Fig 4.1 IxNetwork API documentation link

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.

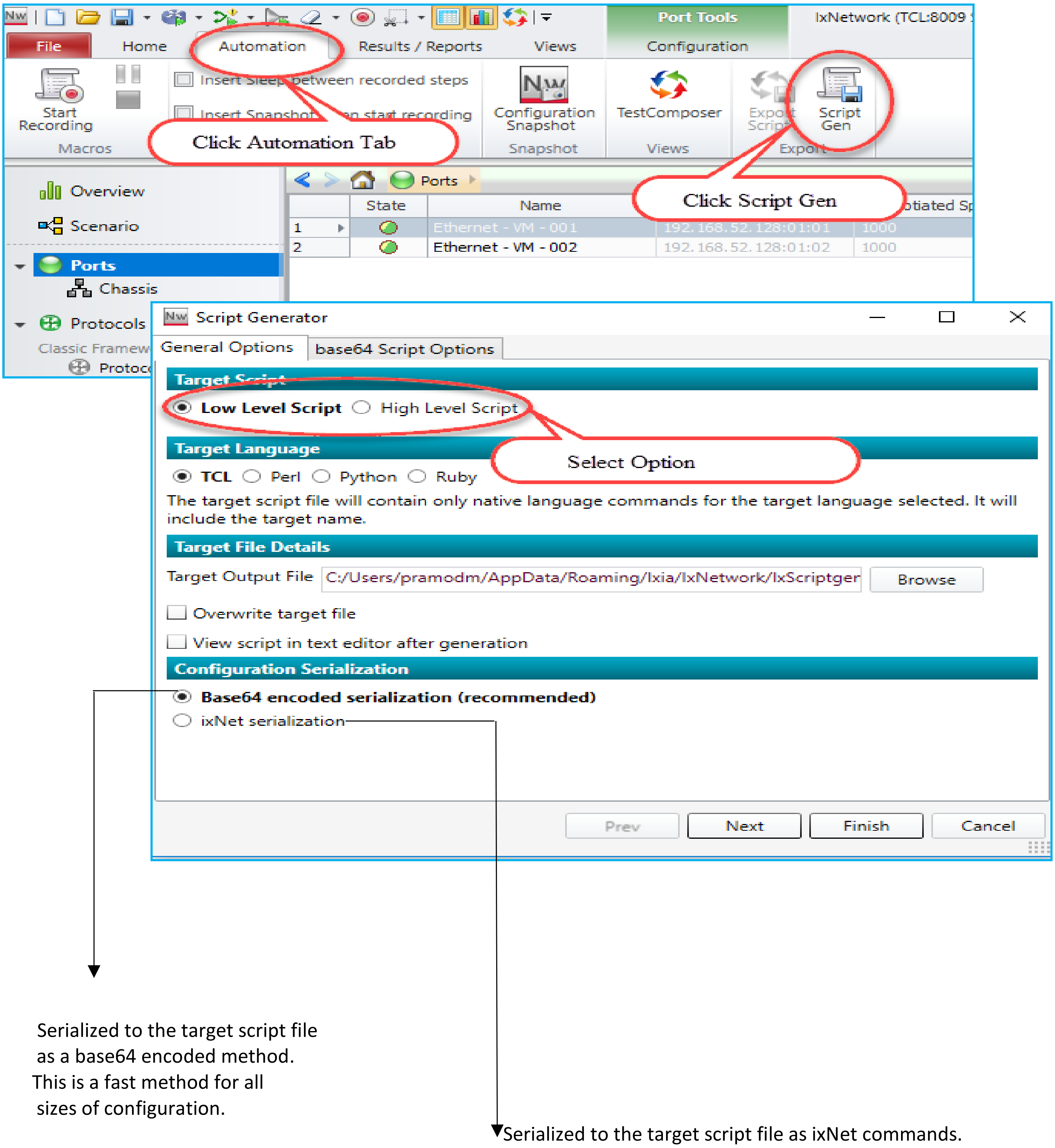


Fig 4.2 IxNetwork ScriptGen link

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.

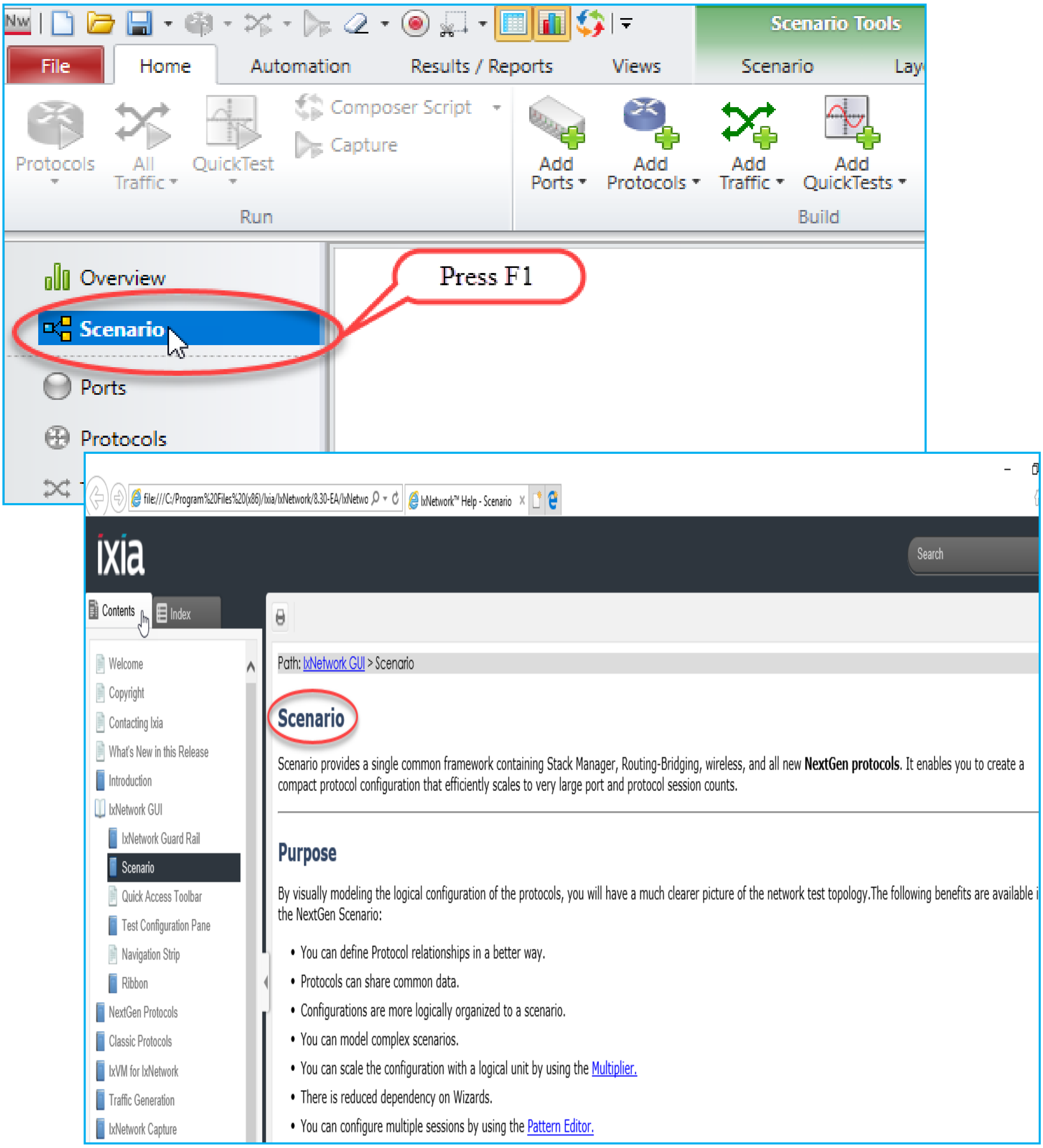


Fig 4.3 IxNetwork F1 option usability

5. To Know More on NGPF:

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

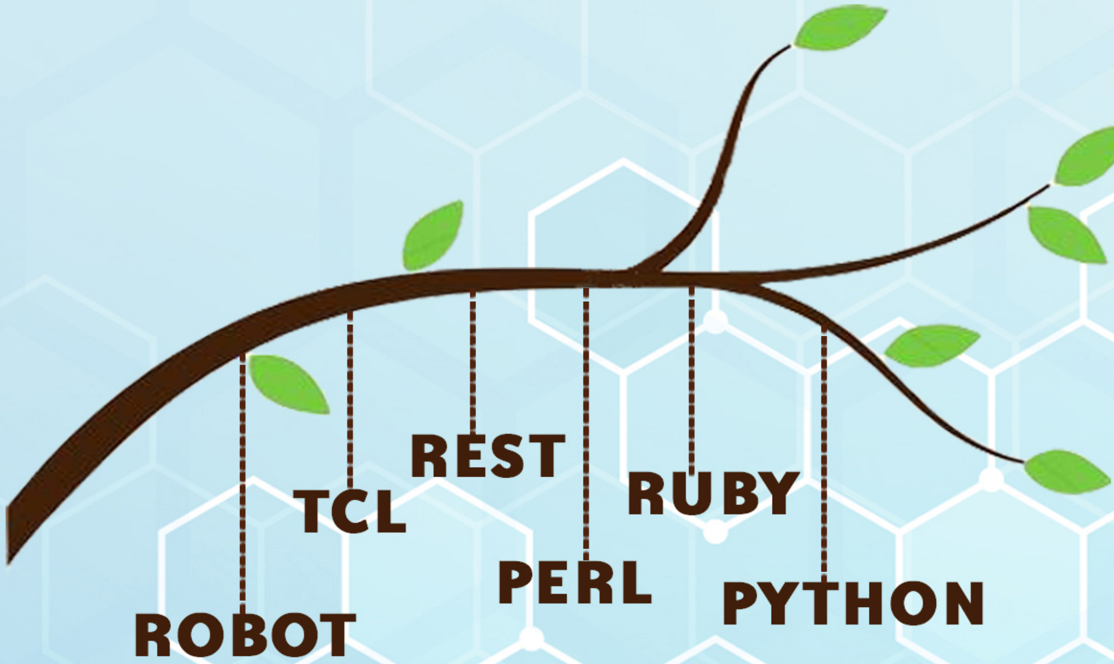
6. Support:

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

ixia | A Keysight Business

AUTOMATION

#WeMakeItEasier



https://github.com/openixia

For queries : support.ix@keysight.com