IxNETWORK REST API   
 QUICK REFERENCE GUIDE

Quick  
Reference  
 Guide

Notices

© Keysight Technologies 2021

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Keysight Technologies, as governed by United States and international copyright laws.

Warranty

**THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED “AS IS,” AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. SHOULD KEYSIGHT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT WILL CONTROL.**

Statement of Compliance.

This product has been designed and tested in accordance with accepted industry standards, and has been supplied in a safe condition. The documentation contains information and warnings that must be followed by the user to ensure safe operation and to maintain the product in a safe condition.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as “Commercial computer software” as defined in DFAR 252.227-7014 (June 1995), or as a “commercial item” as defined in FAR 2.101(a) or as “Restricted computer software” as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Keysight Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

The following general safety precautions must be observed during all phases of operation of this instrument. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the instrument. Keysight Technologies Inc. assumes no liability for the customer’s failure to comply with these requirements.



A CAUTION notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.

  
  
A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Table of Contents

[**1** Overview 1](#_Toc77273431)

[**2** Configure BGP through GUI: 3](#_Toc77273432)

[Add Chassis and Lock Ports: 3](#_Toc77273433)

[Add Topology: 4](#_Toc77273434)

[Emulate a Protocol: 6](#_Toc77273435)

[Device Group Multiplier: 7](#_Toc77273436)

[Edit Protocol Grid: 7](#_Toc77273437)

[Configure BGP: 7](#_Toc77273438)

[Add Network Group: 8](#_Toc77273439)

[Start Protocols: 9](#_Toc77273440)

[Configure Traffic: 9](#_Toc77273441)

[Add Endpoints to Traffic: 10](#_Toc77273442)

[Edit Packet and Setup Flow Groups: 11](#_Toc77273443)

[Setup Frame Size and Rate: 12](#_Toc77273444)

[Setup Flow Tracking and Protocol Behavior: 13](#_Toc77273445)

[Validate Traffic: 14](#_Toc77273446)

[Apply Traffic, Start Traffic, and Statistics View: 15](#_Toc77273447)

[**3** Configure BGP through Automation (RESTPy): 16](#_Toc77273448)

[Initialize Environment: 16](#_Toc77273449)

[Connect to IxNetwork API server: 16](#_Toc77273450)

[Add Chassis and Lock Ports: 16](#_Toc77273451)

[Create Topology: 18](#_Toc77273452)

[Create Device Group: 19](#_Toc77273453)

[Create Ethernet Stack: 20](#_Toc77273454)

[Create Ipv4 Stack: 22](#_Toc77273455)

[Create BGP: 23](#_Toc77273456)

[Create Network Group: 27](#_Toc77273457)

[Start Protocols: 29](#_Toc77273458)

[Configure Traffic: 29](#_Toc77273459)

[Start Traffic and Stop Traffic: 31](#_Toc77273460)

[Disconnect: 32](#_Toc77273461)

[**4** Using find() to query any Object: 33](#_Toc77273462)

[**5** Configure Multi values through RESTPy: 34](#_Toc77273463)

[Get IPv4 Object from configuration using find(): 34](#_Toc77273464)

[Modify IPv4 Address with Single Value: 35](#_Toc77273465)

[Modify IPv4 Address with Increment Value: 35](#_Toc77273466)

[Modify IPv4 Address with Decrement Value: 36](#_Toc77273467)

[Modify IPv4 Address with Random Multivalue: 36](#_Toc77273468)

[Modify IPv4 Address with Repeatable Random Range of Multivalue: 37](#_Toc77273469)

[Modify IPv4 Address with Repeatable Random Mask of Multivalue: 37](#_Toc77273470)

[Modify IPv4 Address with custom Multivalue: 38](#_Toc77273471)

[Modify IPv4 Address with ValueList Multivalue: 38](#_Toc77273472)

[**6** To Know More on RESTPy: 39](#_Toc77273473)

[**7** Support: 40](#_Toc77273474)

[Service and Support 41](#_Toc77273475)

# Overview

* REST Stands for Representational State Transfer (REST)
* RESTPy is a client-side library written in Python for automating IxNetwork using REST APIs.

Benefits of RESTPy:

* RESTPy library comes with all the Python classes and attributes that reflect the IxNetwork API tree.
* Multivalue configuration is made easy.
* RESTPy will verify status codes and handle errors by raising exceptions.
* Eliminates doing loops looking for objects.

Prerequisites:

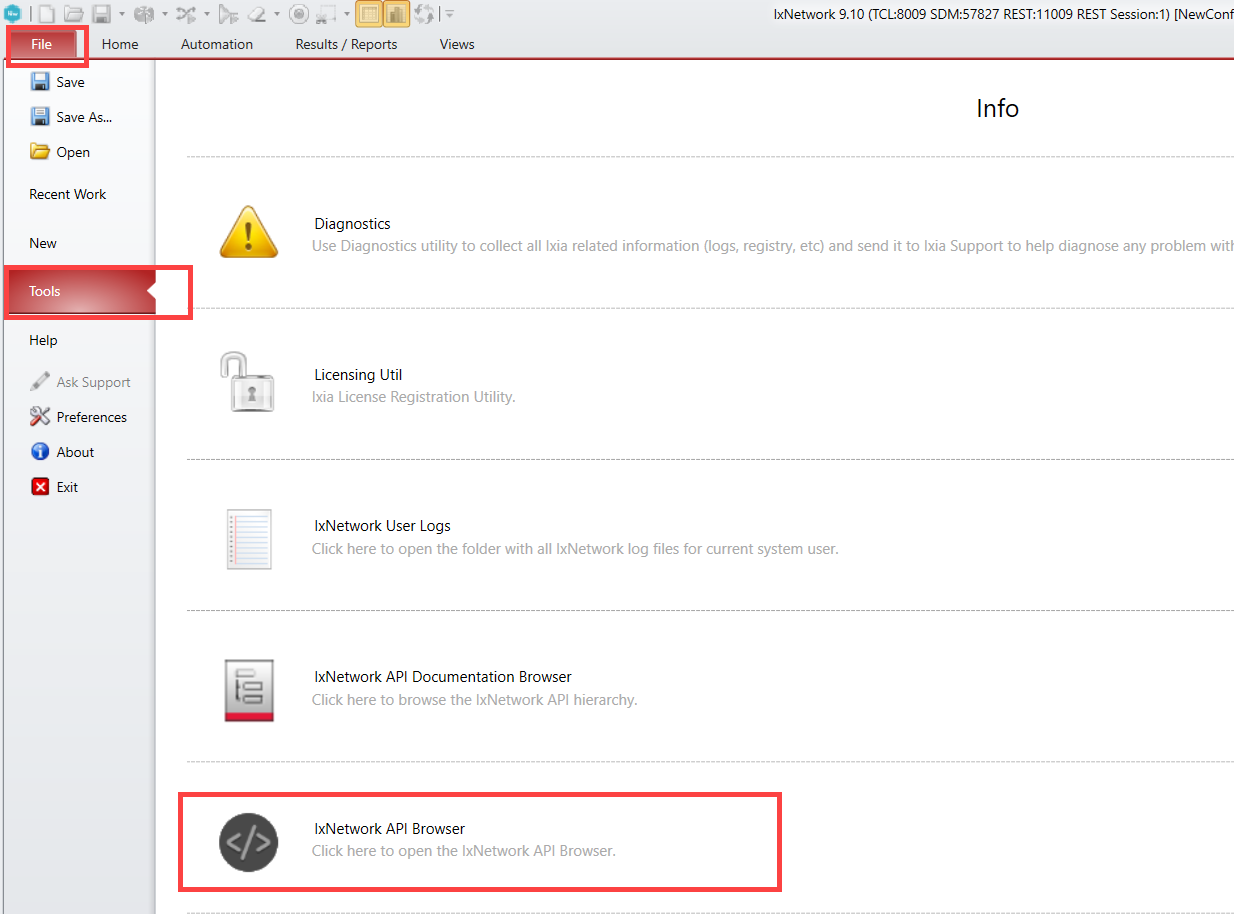
* Prior experience in Python and REST calls will give RESTPy a better understanding.
* Understanding IxNetwork API browser and the mappings between RESTPy and IxNetwork API browser.

Supported Platforms:

* Windows and Linux. IxNetwork NGPF and Classic Framework.

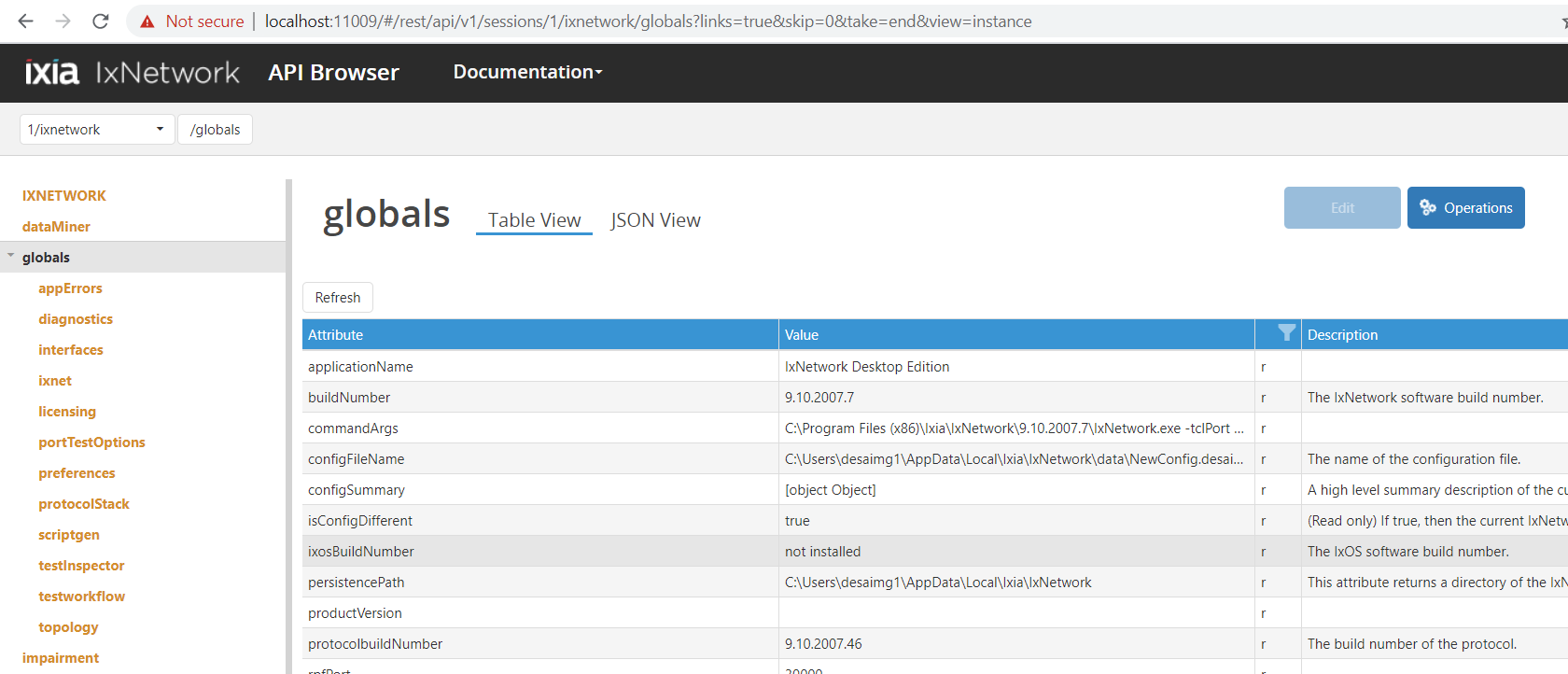
IxNetwork API Browser:

* The API Browser is the entire IxNetwork API tree
* For Windows IxNetwork GUI:
  + Go to File, Tools and select IxNetwork API Browser
  + On a browser: https://<ip>:11009



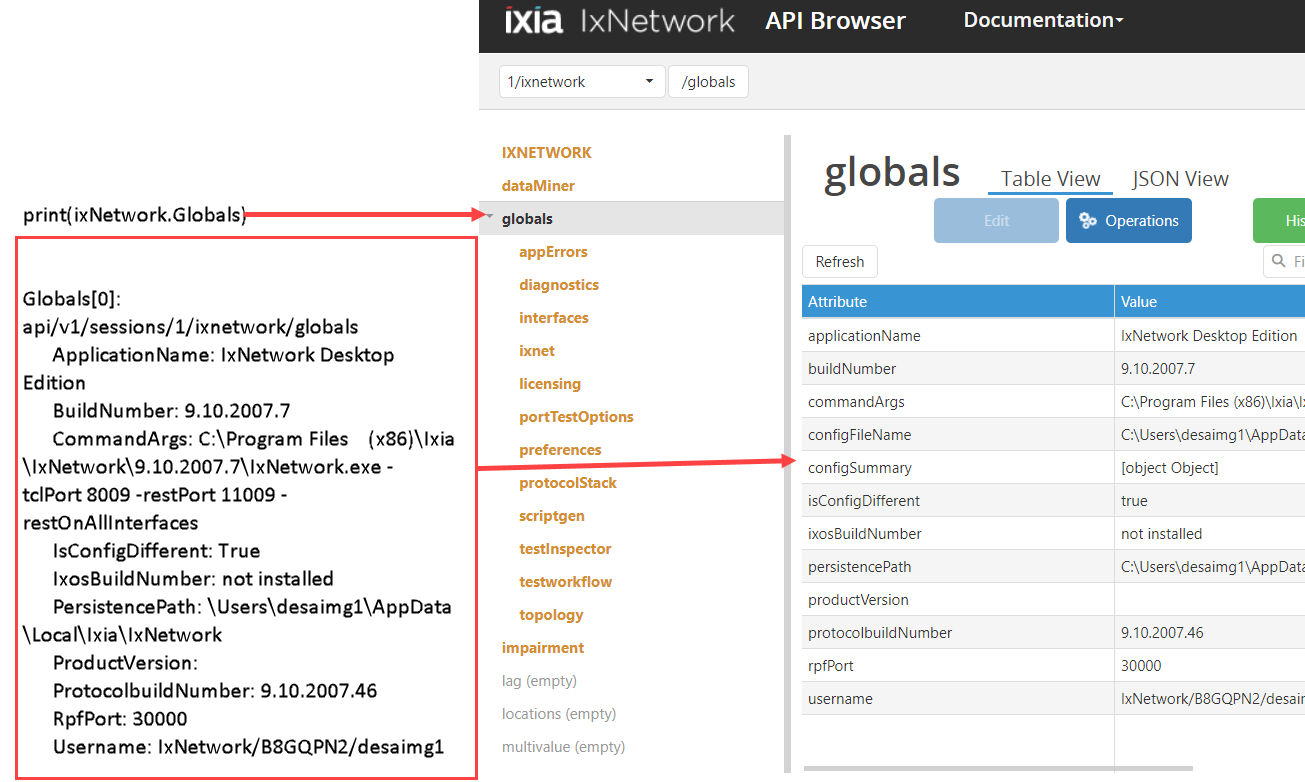
To Open IxNetwork API Browser in Windows

* From web UI:
  + Enter a session and click on the </> icon.
* The left pane in the API browser shows all the API entry nodes.



IxNetwork API Browser

RESTPy Mapping with API Browser



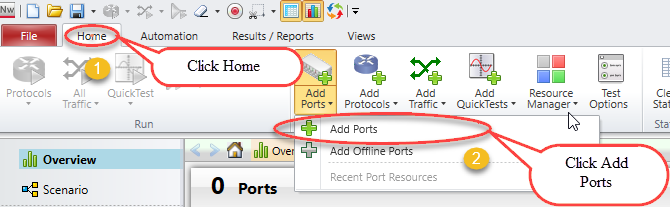
Restpy mapping with IxNetwork API Browser

# Configure BGP through GUI:

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

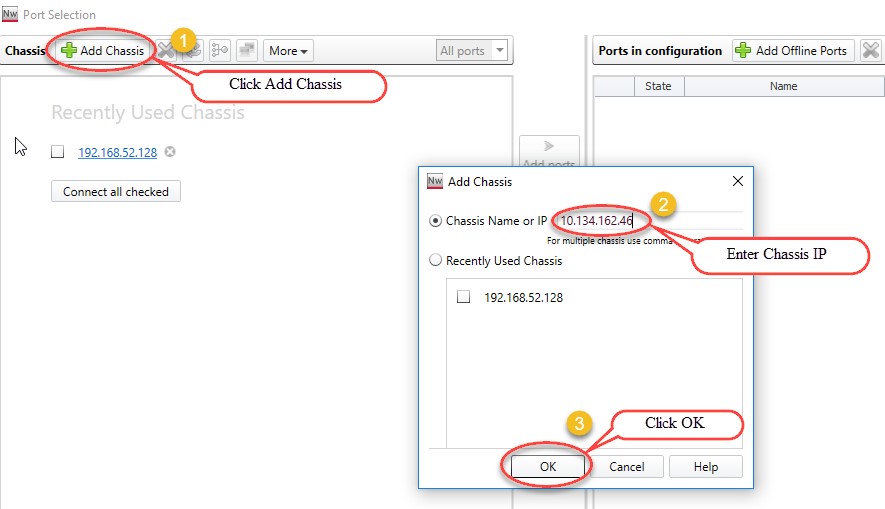
## Add Chassis and Lock Ports:

* The Port Selection window allows you to manage the ports.

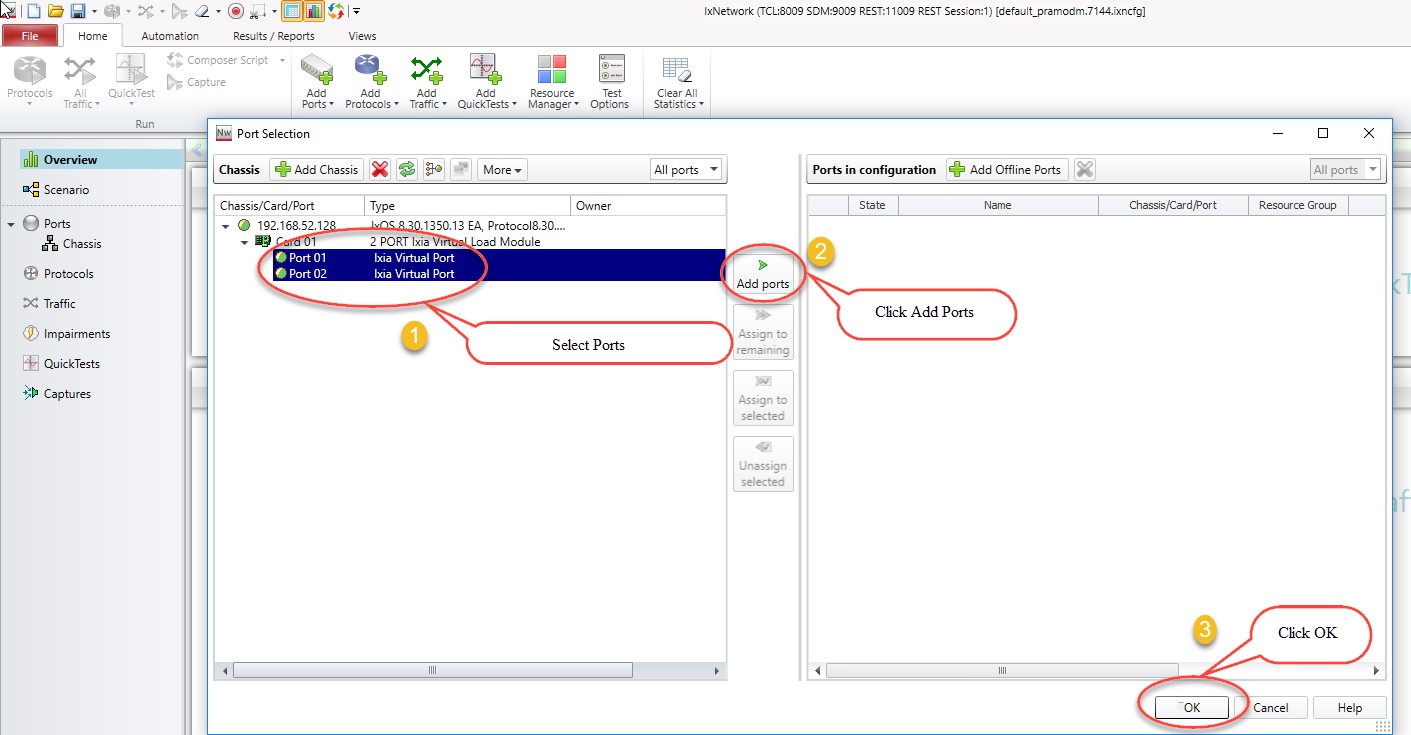


Selecting Ports

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



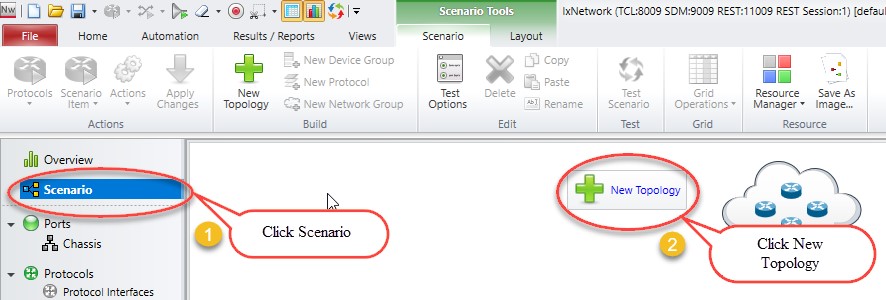
Adding chassis IP



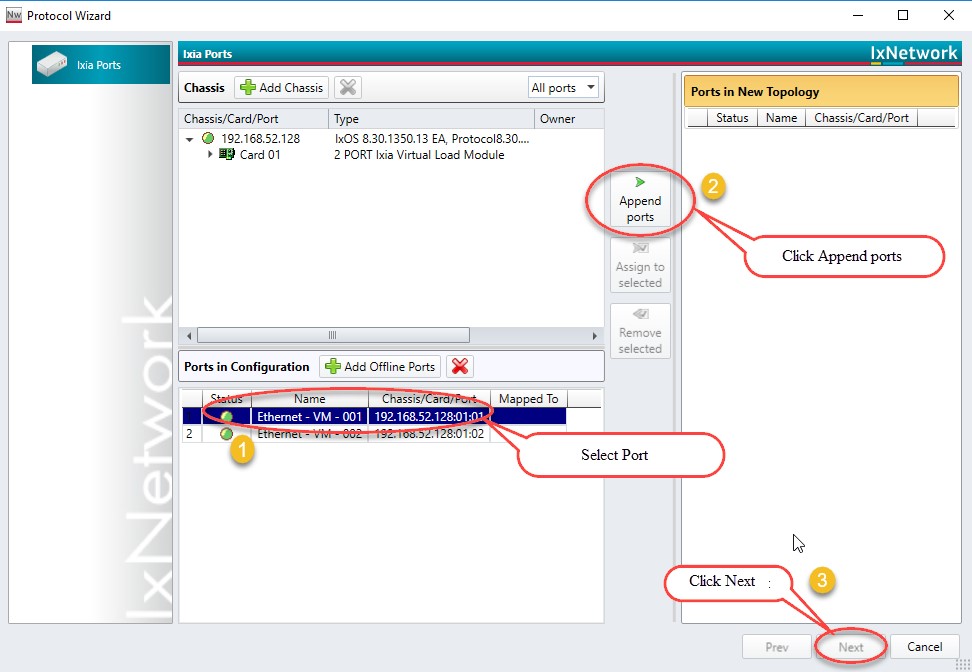
Port selection

## 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 Topology is bound to a virtual port and that virtual port, in turn, binds the physical port.



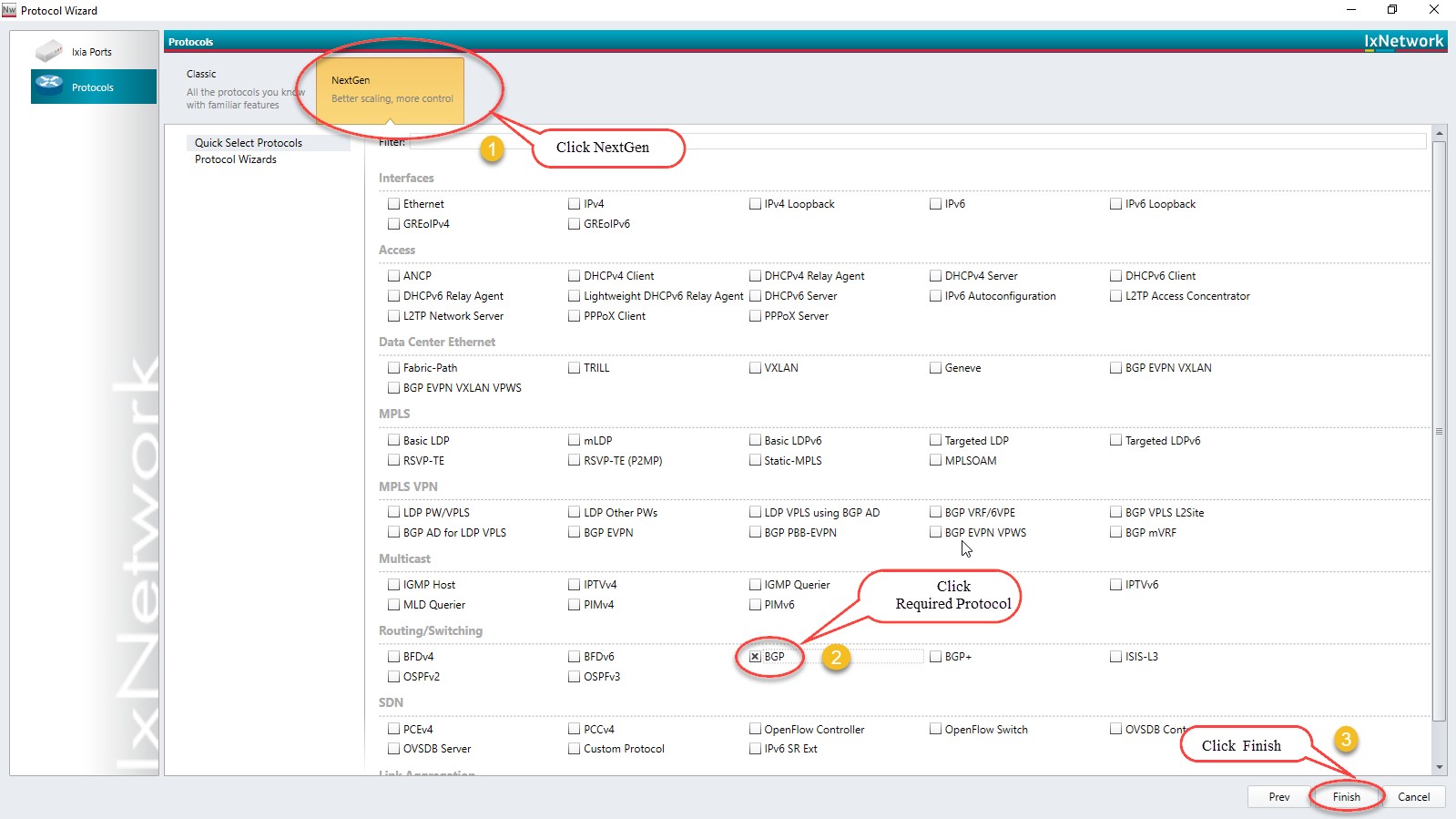
Adding New Topology



Topology with selected ports

## Emulate a Protocol:

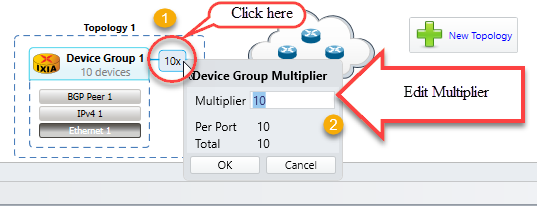
* The Protocols page in the Protocol Wizard allows you to select the protocols in the Topology.
  + The Protocols page lists the available Classic and NextGen protocols under the respective tabs.
  + Click NextGen, and then select the required protocol for the test.
* Presents all supported protocols in Next Generation Protocol Framework in a single window.



Selected Protocol BGP

## 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.

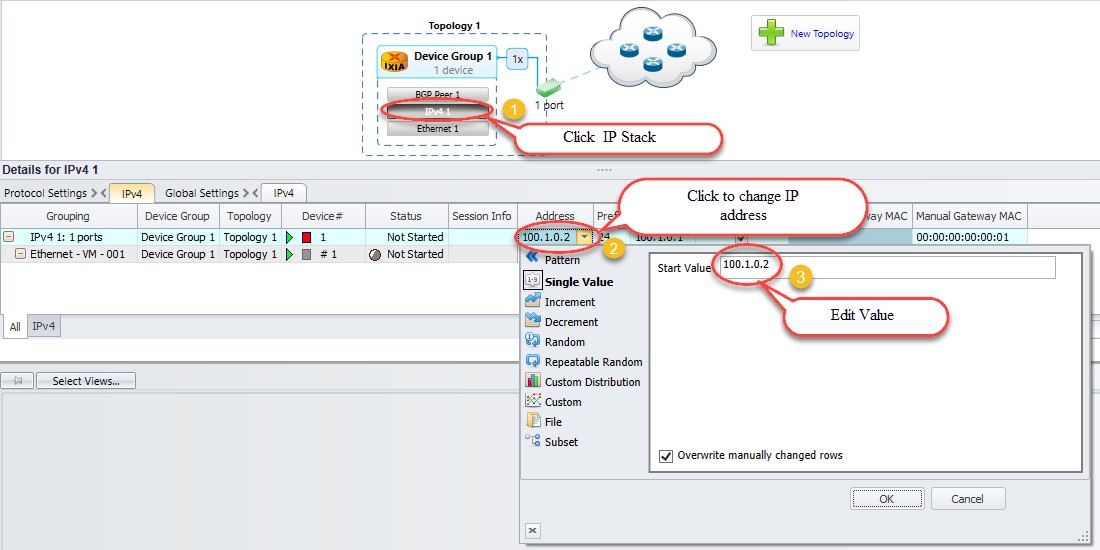


Emulate number of devices by using device group multiplier

## Edit Protocol Grid:

The protocol stack shown in the Scenario view is interactive. Click any protocol stack and edit the values according to the requirement.

**Fig 2.5**



Configuring interfaces

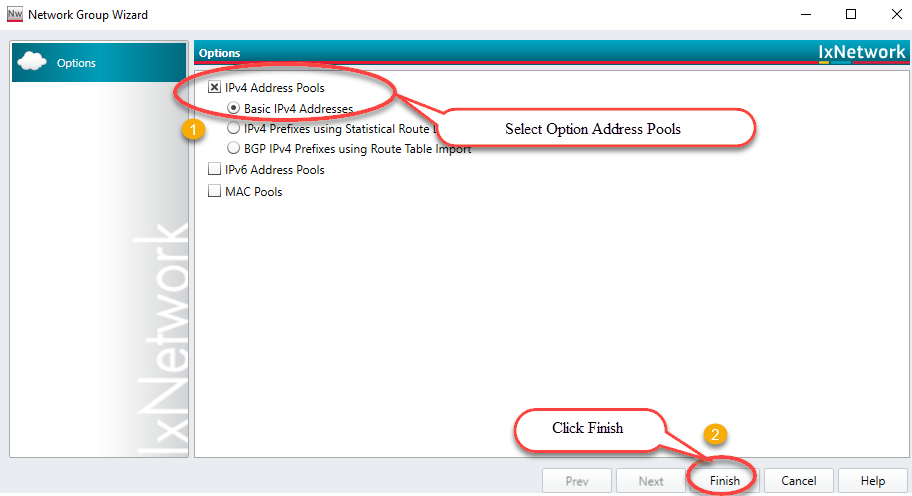
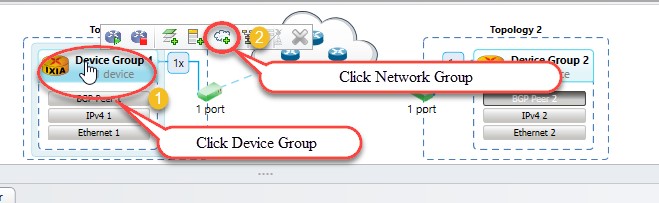
## Configure BGP:

* 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.

## 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.

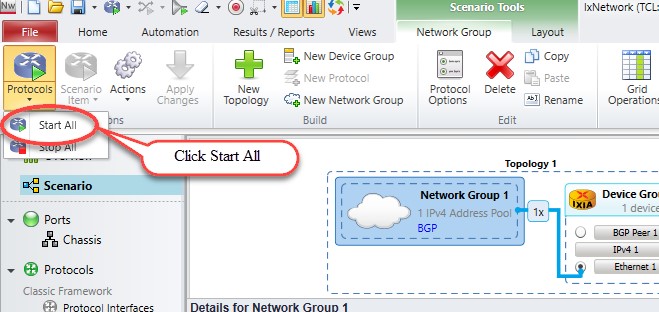
All Devices connected to a Network Group must belong to one of the networks modeled by that Network Group.



Route Profile addition by using network groups

## Start Protocols:

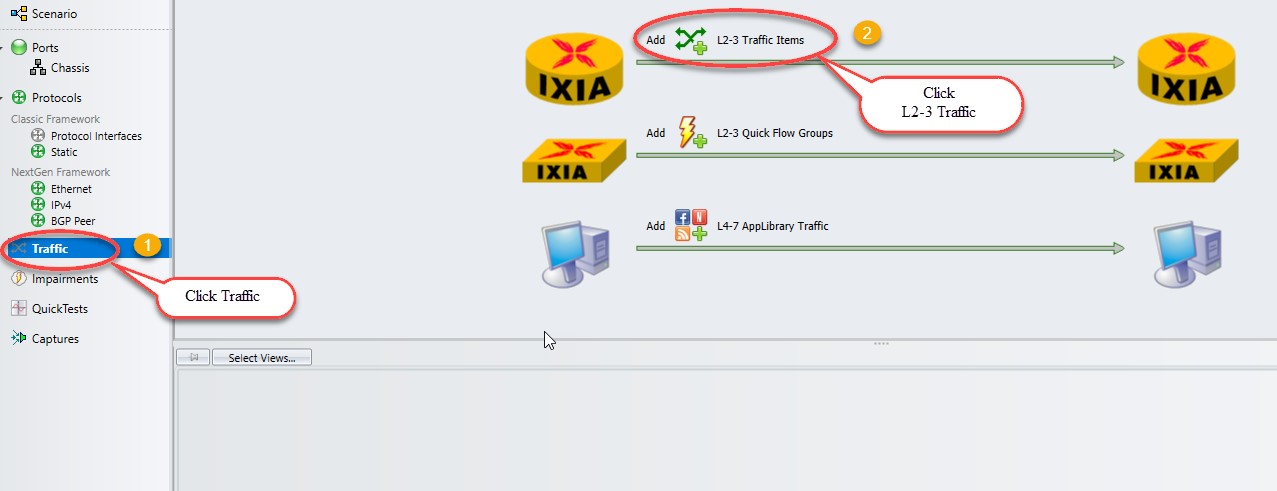
Click Start All to start all the protocols configured in the test.



Brings up all protocol stacks

## Configure Traffic:

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



Configures L2-3 traffic item

## Add Endpoints to Traffic:

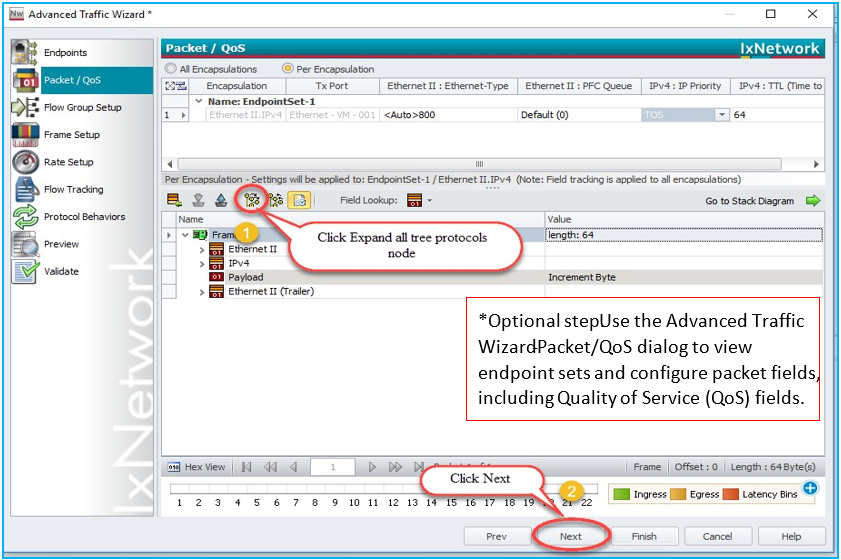
* The Endpoints dialog box is the first dialog box in a series that form the Advanced Traffic Wizard. To access the Endpoints dialog box, click the Endpoints tab in the left pane of the Advanced Wizard window.
* The Endpoints dialog box shows the options to select the traffic endpoints.

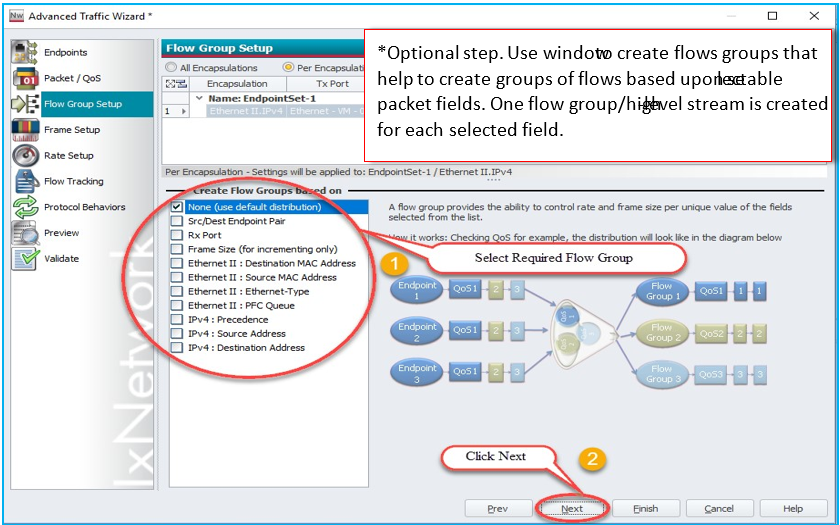


Configures source and destinations endpoints set

## Edit Packet and Setup Flow Groups:

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

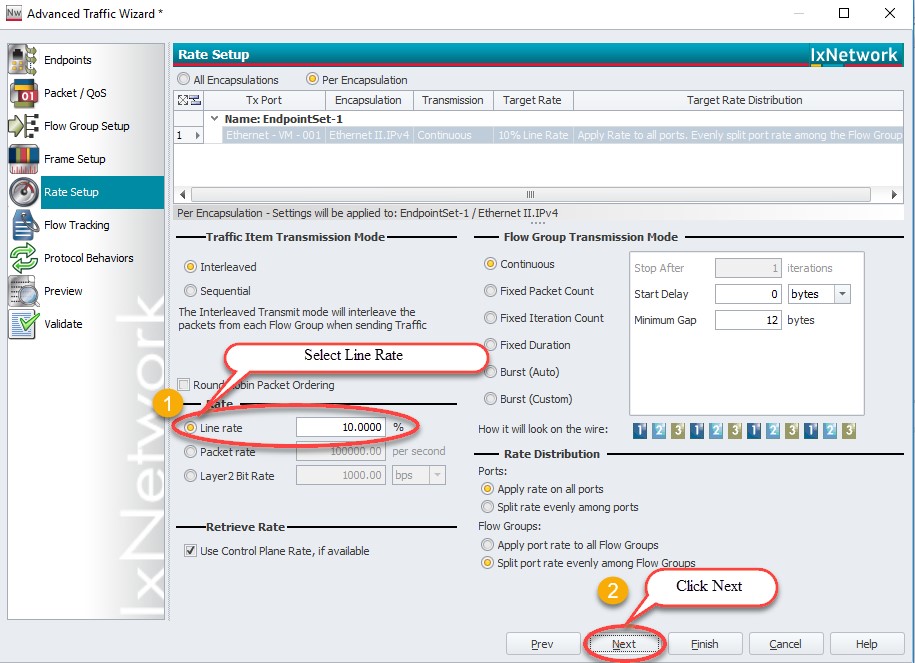
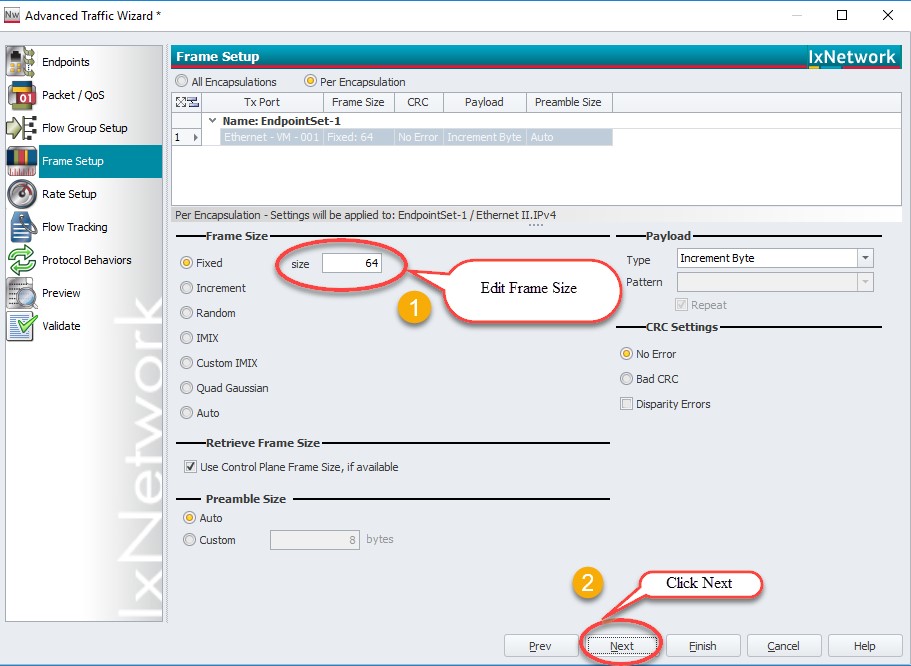




Customizing the packet and creating flow groups

## Setup Frame Size and Rate:

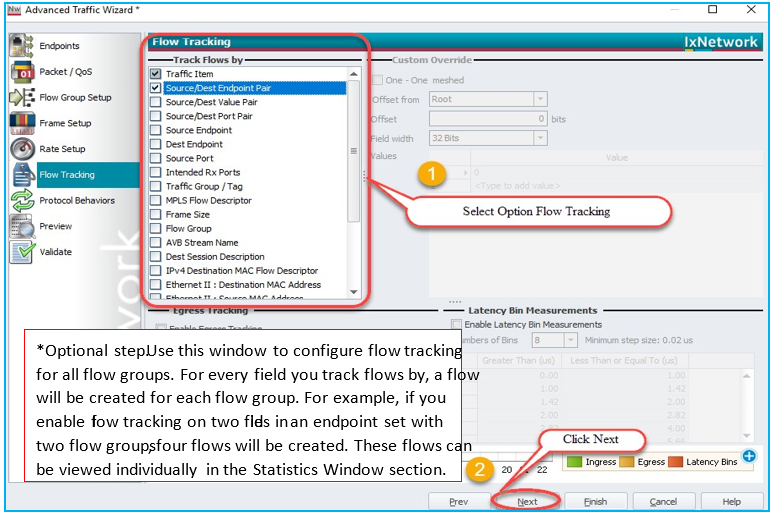
* Setting up the frame Size and Line rate is optional.

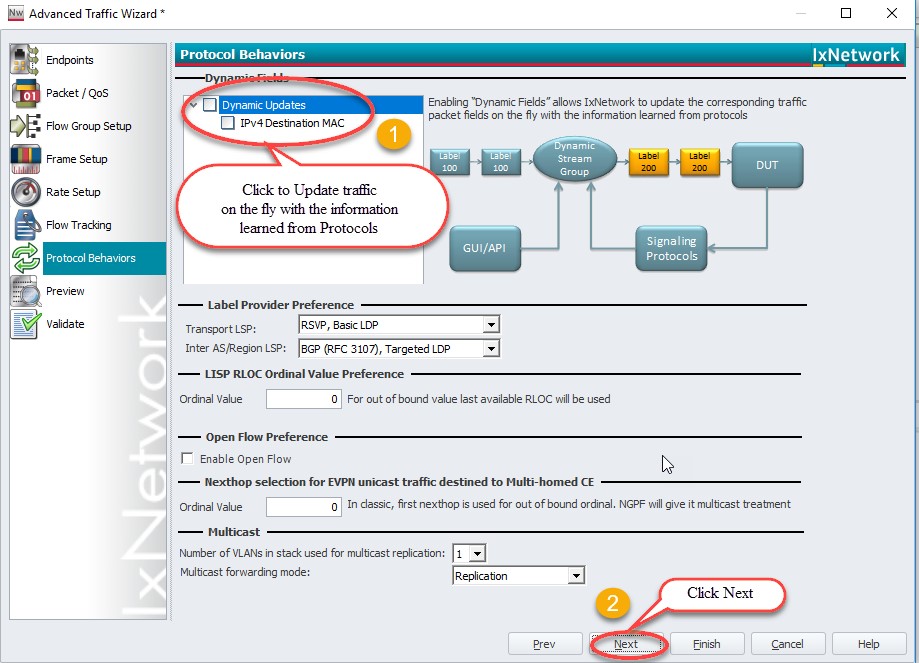


Setting up the frame size and line rate of the traffic

## Setup Flow Tracking and Protocol Behavior:

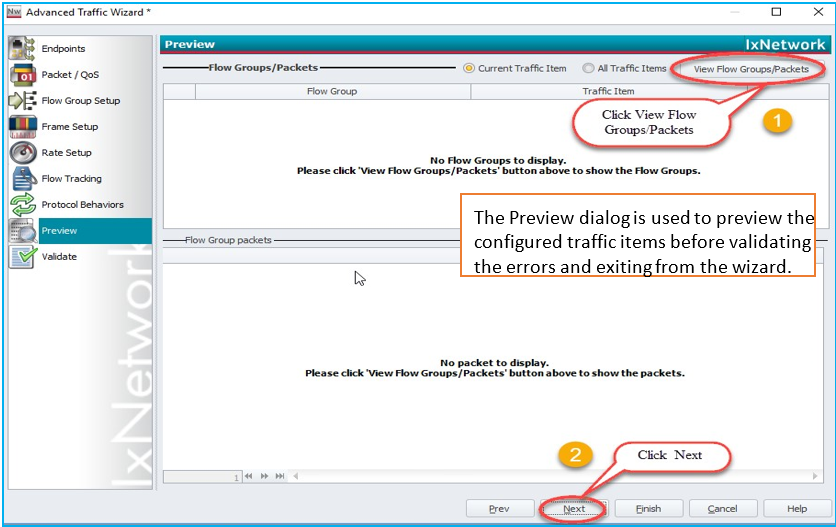
* Setting up the flow tracking and Protocol Behavior is optional.

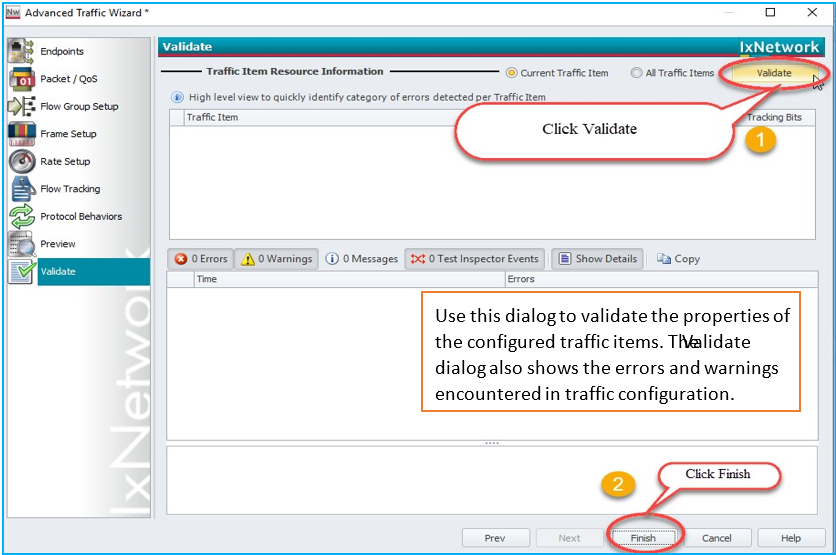




Setting up the flow tracks and traffic update option

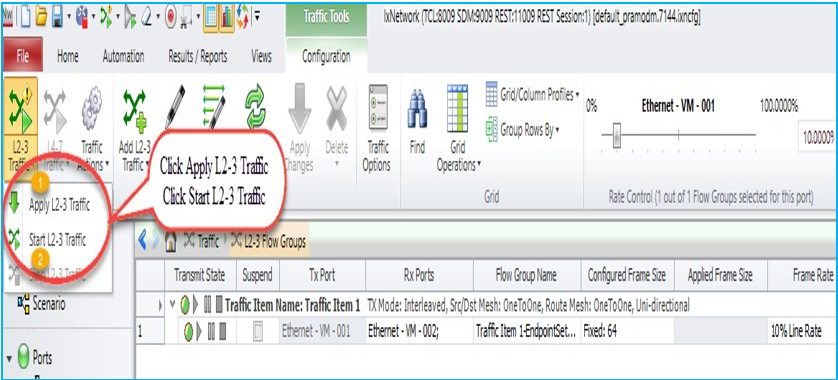
## Validate Traffic:



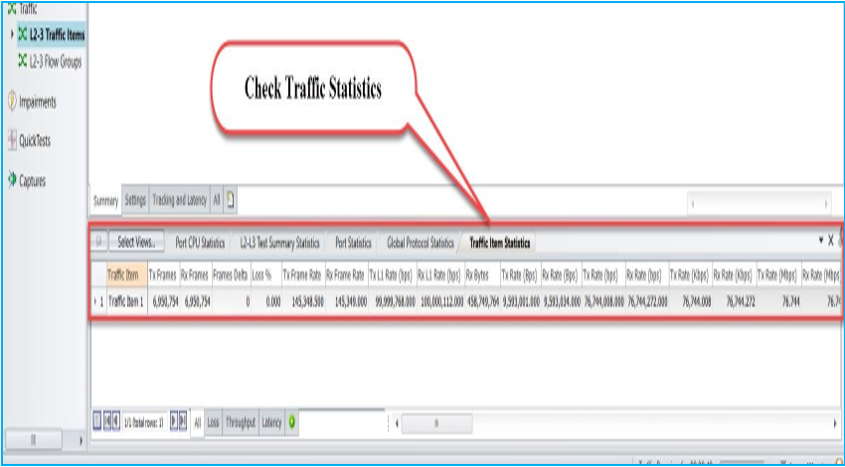


Viewing the flow groups and validating the traffic

## Apply Traffic, Start Traffic, and Statistics View:



Applying and Starting L2-3 Traffic items



Check for traffic item statistics

# Configure BGP through Automation (RESTPy):

This section provides a walk-through to reproduce the BGP emulation scenario by using RESTPy Library.

## Initialize Environment:

Requirement:

* Python2.7 and 3+ versions
* IxNetwork 9.0 and above versions

Install RESTPy Packages “pip install ixnetwork-restpy”

## Connect to IxNetwork API server:

from ixnetwork\_restpy import SessionAssistant

# As the initial step it needs connecting with API server. In this case API server is local machine so, we are using 127.0.0.1 IP. ClearConfig should be True to clear existing configurations in IxNetwork.

*session\_assistant = SessionAssistant(IpAddress='127.0.0.1',*

*UserName='admin',*

*Password='admin',*

*LogLevel=SessionAssistant.LOGLEVEL\_INFO,*

*ClearConfig=True)*

ixnetwork = session\_assistant.Ixnetwork

# ixnetwork is a session Object which we are going to use further. UserName and Password parameters are required for Linux based api server

## Add Chassis and Lock Ports:

*#To add chassis*

*chassisIp = "10.39.68.149"*

*ixnetwork.AvailableHardware.Chassis.add(Hostname=chassisIp)*

*# Before adding ports, we need to create vport. We are adding two ports. So, we are creating two vport object*

*vport1 = ixnetwork.Vport.add()*

*vport2 = ixnetwork.Vport.add()*

*vportList = [vport1, vport2]*

*print(vportList)*

*[<ixnetwork\_restpy.testplatform.sessions.ixnetwork.vport.vport.Vport object at 0x03D34490>, <ixnetwork\_restpy.testplatform.sessions.ixnetwork.vport.vport.Vport object at 0x00D0DFD0>]*

*Map vports to physical ports:*

*# To assign physical ports with vports, RESTPy takes in the format of dictionary values.*

*For Ex: [{'Arg1': '10.39.68.149', 'Arg2': 1, 'Arg3': 3}, {'Arg1': '10.39.68.149', 'Arg2': 1, 'Arg3': 4}]*

*Here Arg1 is Chassis IP, Arg2 is Slot Number and Arg3 is Port Number*

*portList = [[chassisIp, 1, 3], [chassisIp, 1, 4]]*

*testPorts = []*

*for port in portList:*

*testPorts.append(dict(Arg1=port[0], Arg2=port[1], Arg3=port[2]))*

*# following syntax will take care of Assigning ports*

*Parameter we need for AssignPorts API*

*- Arg1 (list(dict(arg1:str,arg2:str,arg3:str))): A list of chassis, card, port combinations to include.*

*- Arg2 (list(dict(arg1:str,arg2:str,arg3:str))): A list of chassis, card, port combinations to exclude.*

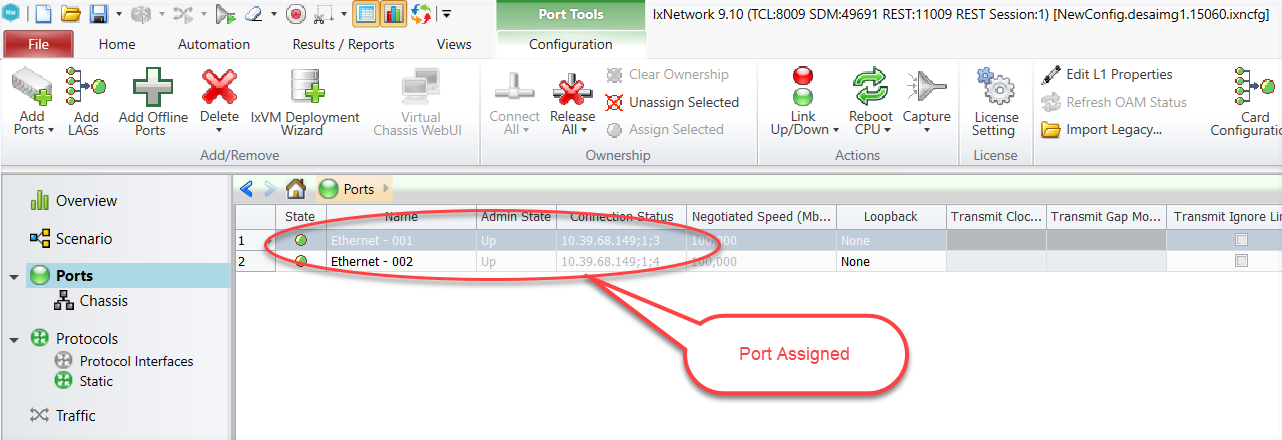
*- Arg3 (list(str[None | /api/v1/sessions/1/ixnetwork/vport])): A list of vport object references*

*that hardware ports will be attached to.*

*- Arg4 (bool): If true, it will clear ownership on the hardware ports.*

*ixnetwork.AssignPorts(testPorts, [], vportList, True)*

*# We have given an empty list as we don’t have any port combinations to exclude*



Chassis connected, and selected ports are assigned

## Create Topology:

*Create BGP-1 Topology:*

*# To create new topology use following methods. To map the topology with port, give vport object we created earlier*

*topoObj1 = ixnetwork.Topology.add(Ports=vport1)*

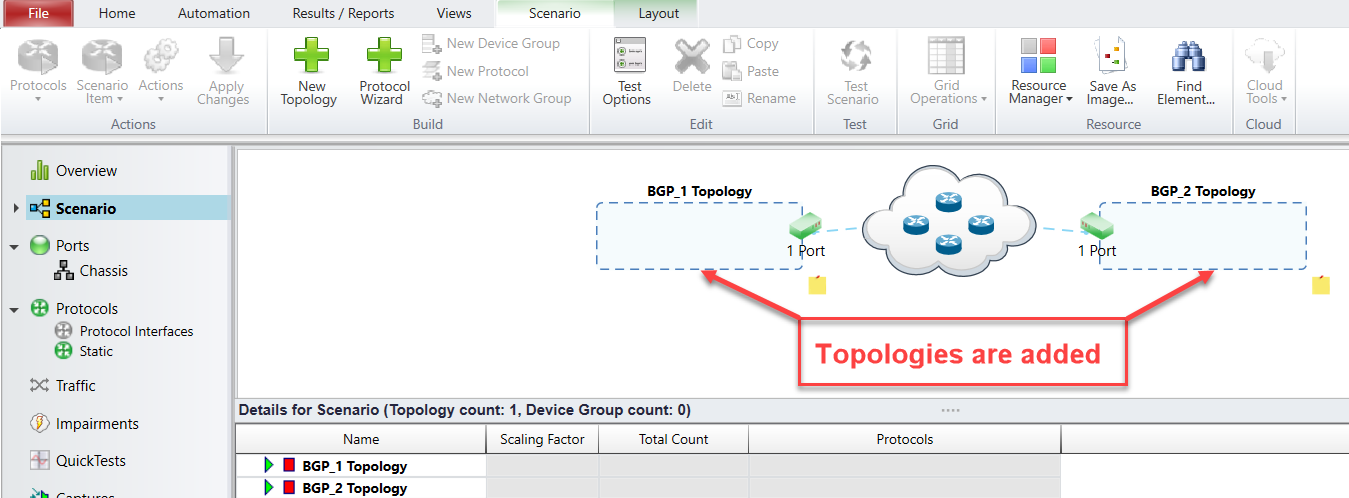
*# To modify name of topology 1*

*topoObj1.Name = “BGP\_1 Topology”*

*Create BGP-2 Topology:*

*# To create new topology with topology name we can use following single line of code*

*topoObj2 = ixnetwork.Topology.add(Ports=vport2, Name = "BGP\_2 Topology")*



Topology are added

## Create Device Group:

*Create BGP-1 Device Group:*

*# To create new device group use DeviceGroup.add() with topology object. Multiplier and Name*

*can also be added in a single line*

*dgObj1 = topoObj1.DeviceGroup.add(Name = "BGP\_1 Device Group", Multiplier=1)*

*# To modify multiplier of device group use devicegroup object and Multiplier attribute*

*dgObj1.Multiplier = 1*

*# To modify the name of device group use devicegroup object and Name attribute*

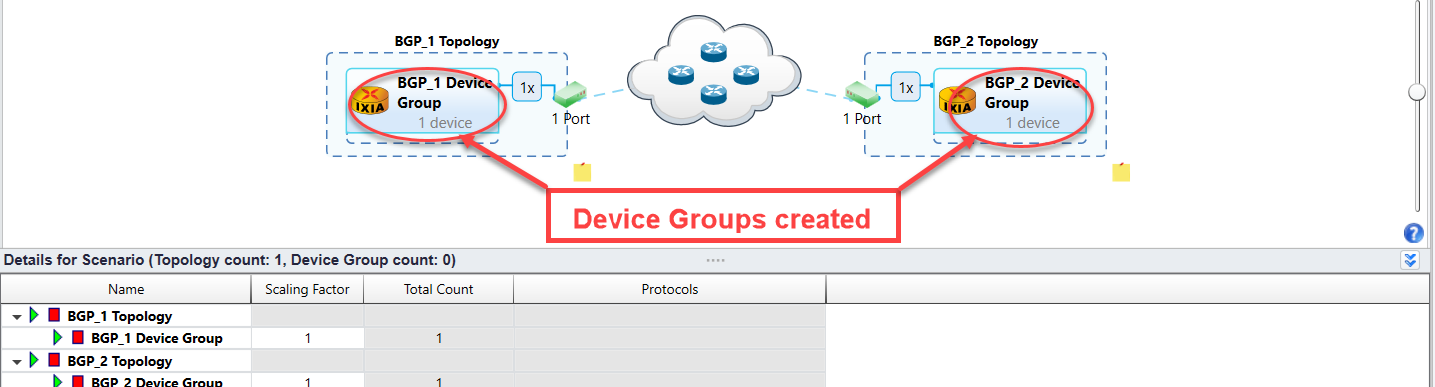
*dgObj1.Name = “BGP\_1 Device Group”*

*Create BGP-2 Device Group:*

*# To create new device group use DeviceGroup.add() with topology object. Multiplier and Name*

*can also be added in a single line*

*dgObj2 = topoObj2.DeviceGroup.add(Name = "BGP\_2 Device Group", Multiplier=1)*



Device groups added to respective topologies

## Create Ethernet Stack:

*Create Ethernet Stack-1 on topology-1 device group 1:*

*# To create a new ethernet stack use DeviceGroup Object*

*ethObj1 = dgObj1.Ethernet.add()*

*# To retrieve an attribute of ethernet object*

*print(ethObj1)*

*Ethernet[0]: /api/v1/sessions/1/ixnetwork/topology/1/deviceGroup/1/ethernet/1*

*ConnectedVia: []*

*Count: 1*

*DescriptiveName: Ethernet 1*

*EnableVlans: /api/v1/sessions/1/ixnetwork/multivalue/5*

*Errors: []*

*Mac: /api/v1/sessions/1/ixnetwork/multivalue/3*

*Mtu: /api/v1/sessions/1/ixnetwork/multivalue/4*

*Multiplier: 1*

*Name: Ethernet 1*

*NotifyMACMove: False*

*SessionStatus: ['notStarted']*

*StackedLayers: []*

*StateCounts: {'arg1': 1, 'arg2': 1, 'arg3': 0, 'arg4': 0}*

*Status: notStarted*

*UseVlans: False*

*VlanCount: 1*

*# To fetch ethernet ‘MAC’*

*print(ethObj1.Mac)*

*Inc: 00:11:01:00:00:01, 00:00:00:00:00:01*

*# To modify ethernet ‘MAC‘ with a single value*

*ethObj1.Mac.Single("11:11:11:11:11:11")*

*# To modify the name of ethernet stack*

*ethObj1.Name = "Eth1"*

*Create Ethernet Stack-1 on topology-2 device group 1:*

*# To create a new ethernet stack*

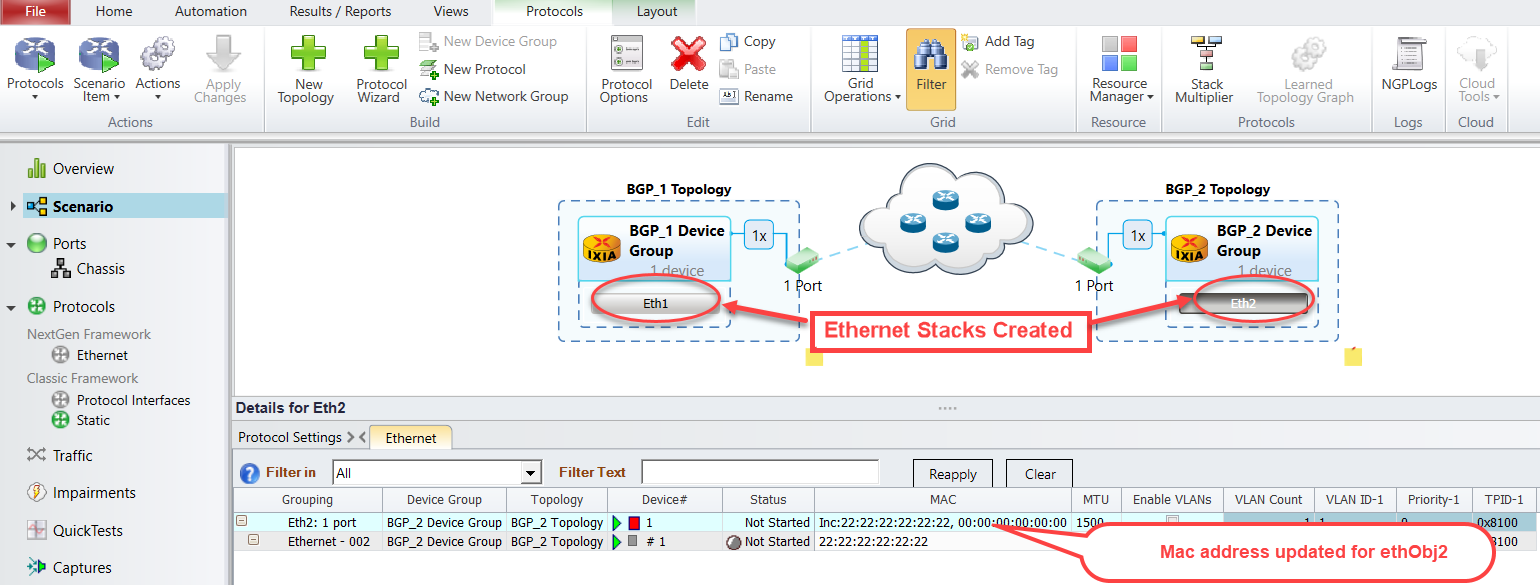
*ethObj2 = dgObj2.Ethernet.add()*

*# To modify ethernet ‘MAC‘ with increment value*

*ethObj2.Mac.Increment(start\_value="22:22:22:22:22:22", step\_value="00:00:00:00:00:00")*

*# To modify the name of ethernet stack*

*ethObj2.Name = "Eth2"*



Ethernet stacks added to device groups

## Create Ipv4 Stack:

*Creating Ipv4 Stack on topology-1 device group 1:*

*# To create a new IPv4 stack use Ethernet Object*

*ipv4Obj1 = ethObj1.Ipv4.add()*

*# To retrieve contents of IPv4 stack*

*print(ipv4Obj1)*

*Ipv4[0]: /api/v1/sessions/1/ixnetwork/topology/1/deviceGroup/1/ethernet/1/ipv4/1*

*Address: /api/v1/sessions/1/ixnetwork/multivalue/9*

*ConnectedVia: []*

*Count: 1*

*DescriptiveName: IPv4 1*

*EnableGratarpRarp: /api/v1/sessions/1/ixnetwork/multivalue/14*

*Errors: []*

*GatewayIp: /api/v1/sessions/1/ixnetwork/multivalue/11*

*Ipv4GratarpRarp: /api/v1/sessions/1/ixnetwork/multivalue/15*

*ManualGatewayMac: /api/v1/sessions/1/ixnetwork/multivalue/13*

*Multiplier: 1*

*Name: IPv4 1*

*Prefix: /api/v1/sessions/1/ixnetwork/multivalue/10*

*ResolveGateway: /api/v1/sessions/1/ixnetwork/multivalue/12*

*ResolvedGatewayMac: ['removePacket[Unresolved]']*

*SessionInfo: ['none']*

*SessionStatus: ['notStarted']*

*StackedLayers: []*

*StateCounts: {'arg1': 1, 'arg2': 1, 'arg3': 0, 'arg4': 0}*

*Status: notStarted*

*# To modify IPv4 Stack Name*

*ipv4Obj1.Name = "IPv41"*

*# To modify the IPv4 'address'*

*ipv4Obj1.Address.Single("20.0.0.1")*

*# To modify the IPv4 ‘Gateway address'*

*ipv4Obj1.GatewayIp.Single("20.0.0.2")*

*Creating Ipv4 Stack on topology-2 device group 1:*

*# To create new IPv4 stack use*

*ipv4Obj2=ethObj2.Ipv4.add()*

*# To modify IPv4 ‘Stack Name’*

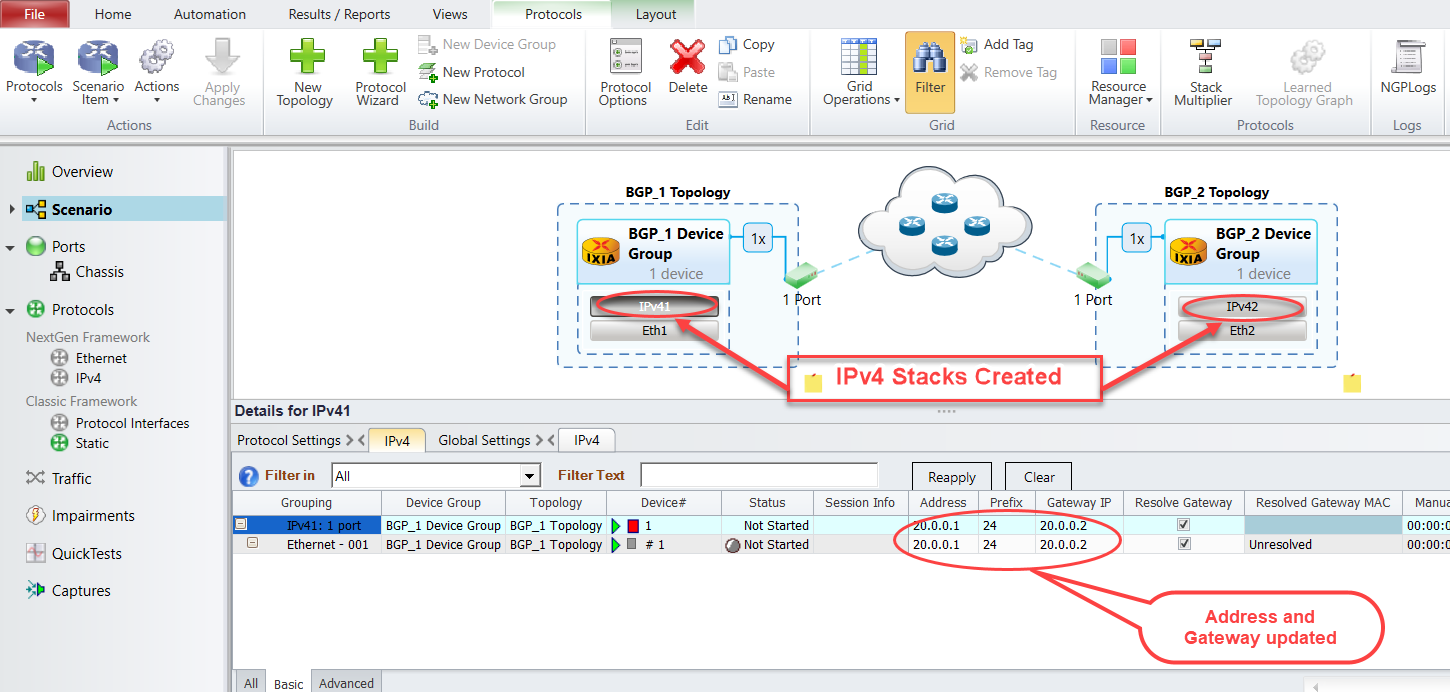
*ipv4Obj2.Name='IPv42'*

*# To modify the IPv4 'address' use Ipv4 Object and Address Attribute*

*ipv4Obj2.Address.Single("20.0.0.2")*

*# To modify the IPv4 'gatewayIp' use Ipv4 Object and GatewayIp Attribute*

*ipv4Obj2.GatewayIp.Single("20.0.0.1")*



IPv4 stacks added to ethernet stacks

## Create BGP:

*Create BGP on topology-1 device group 1:*

*# To create a new BgpIPv4 protocol. The name denotes stack name*

*bgpObj1 = ipv4Obj1.BgpIpv4Peer.add(Name="BGP1")*

*# To retrieve attributes of bgpObj1*

*print(bgpObj1)*

*BgpIpv4Peer[0]: /api/v1/sessions/1/ixnetwork/topology/1/deviceGroup/1/ethernet/1/ipv4/1/bgpIpv4Peer/1*

*ActAsRestarted: /api/v1/sessions/1/ixnetwork/multivalue/40*

*Active: /api/v1/sessions/1/ixnetwork/multivalue/121*

*AdvertiseEndOfRib: /api/v1/sessions/1/ixnetwork/multivalue/41*

*AdvertiseEvpnRoutesForOtherVtep: False*

*AdvertiseTunnelEncapsulationExtendedCommunity: /api/v1/sessions/1/ixnetwork/multivalue/29*

*AlwaysIncludeTunnelEncExtCommunity: /api/v1/sessions/1/ixnetwork/multivalue/42*

*AsSetMode: /api/v1/sessions/1/ixnetwork/multivalue/33*

*Authentication: /api/v1/sessions/1/ixnetwork/multivalue/51*

*BgpFsmState: ['none']*

*BgpId: /api/v1/sessions/1/ixnetwork/multivalue/25*

*BgpLsAsSetMode: /api/v1/sessions/1/ixnetwork/multivalue/119*

*BgpLsEnableAsPathSegments: /api/v1/sessions/1/ixnetwork/multivalue/120*

*BgpLsEnableCluster: /api/v1/sessions/1/ixnetwork/multivalue/117*

*BgpLsEnableExtendedCommunity: /api/v1/sessions/1/ixnetwork/multivalue/116*

*BgpLsNoOfASPathSegments: 1*

*BgpLsNoOfClusters: 1*

*BgpLsNoOfCommunities: 1*

*BgpLsOverridePeerAsSetMode: /api/v1/sessions/1/ixnetwork/multivalue/118*

*CapabilityIpV4Mdt: /api/v1/sessions/1/ixnetwork/multivalue/91*

*CapabilityIpV4Mpls: /api/v1/sessions/1/ixnetwork/multivalue/85*

*CapabilityIpV4MplsVpn: /api/v1/sessions/1/ixnetwork/multivalue/86*

*CapabilityIpV4Multicast: /api/v1/sessions/1/ixnetwork/multivalue/84*

*CapabilityIpV4MulticastVpn: /api/v1/sessions/1/ixnetwork/multivalue/93*

*CapabilityIpV4Unicast: /api/v1/sessions/1/ixnetwork/multivalue/83*

*.*

*.*

*.*

*.*

*[truncated]*

*# To modify the BgpIpv4 "enableBgpId" use Bgp Object and EnableBgpId Attribute*

*bgpObj1.EnableBgpId.Single(True)*

*# To modify Bgp Stack name use Bgp Object and Name Attribute*

*bgpObj1.Name='Topo1Bgp'*

*# To modify BgpIPv4 ‘holdTimer’*

*bgpObj1.HoldTimer.Single(90)*

*# To modify BgpIPv4 ‘dutIp’*

*bgpObj1.DutIp.Single('20.0.0.2')*

*# To modify BgpIPv4 ‘enableGracefulRestart’*

*bgpObj1.EnableGracefulRestart.Single(True)*

*# To modify BgpIPv4 ‘restartTime’*

*bgpObj1.RestartTime.Single(45)*

*# To modify BgpIPv4 ‘type’*

*bgpObj1.Type.Single('internal')*

*# To modify BgpIPv4 ‘enableBgpIdSameasRouterId’*

*bgpObj1.EnableBgpIdSameasRouterId.Single(True)*

*# To modify BgpIPv4 ‘staleTime’*

*bgpObj1.StaleTime.Single(0)*

*# To modify BgpIPv4 ‘flap’*

*bgpObj1.Flap.Single(False)*

*Create BGP on topology-2 device group1:*

*# To create a new BgpIPv4 protocol*

*bgpObj2 = ipv4Obj2.BgpIpv4Peer.add(Name="BGP2")*

*# To modify the BgpIpv4 "enableBgpId"*

*bgpObj2.EnableBgpId.Single(True)*

*# To modify ‘bgpIpv4Peer’ name*

*bgpObj2.Name='Topo2Bgp'*

*# To modify BgpIPv4 ‘holdTimer’*

*bgpObj2.HoldTimer.Single(90)*

*# To modify BgpIPv4 ‘dutIp’*

*bgpObj2.DutIp.Single('20.0.0.1')*

*# To modify BgpIPv4 ‘enableGracefulRestart’*

*bgpObj2.EnableGracefulRestart.Single(True)*

*# To modify BgpIPv4 ‘restartTime’*

*bgpObj2.RestartTime.Single(45)*

*# To modify BgpIPv4 ‘type’*

*bgpObj2.Type.Single('internal')*

*# To modify BgpIPv4 ‘enableBgpIdSameasRouterId’*

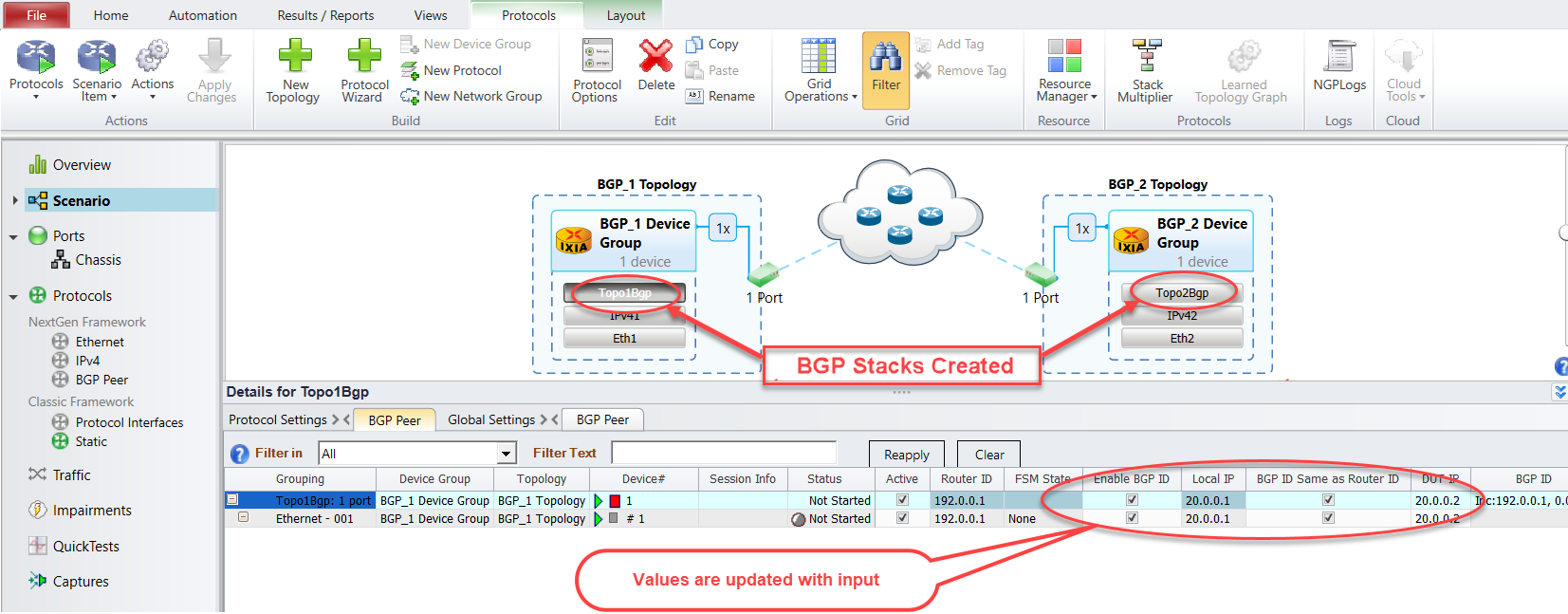
*bgpObj2.EnableBgpIdSameasRouterId.Single(True)*

*# To modify BgpIPv4 ‘staleTime’*

*bgpObj2.StaleTime.Single(0)*

*# To modify BgpIPv4 ‘flap’*

*bgpObj2.Flap.Single(False)*



BGP stacks added to ipv4 stacks

## Create Network Group:

*Create Network Group on topology-1 device group1:*

*# To create a new network group*

*nwGrpObj1 = dgObj1.NetworkGroup.add()*

*# To modify the name of the network group*

*nwGrpObj1.Name = "BGP\_NW\_GRP1"*

*# To modify the multiplier of the network group*

*nwGrpObj1.Multiplier = 5*

*# To create IPv4 Prefix Pool for network group*

*prefixObj1 = nwGrpObj1.Ipv4PrefixPools.add()*

*# To retrieve details of configured IPv4 Prefix Pool*

*print(prefixObj1)*

*Ipv4PrefixPools[0]: /api/v1/sessions/1/ixnetwork/topology/1/deviceGroup/1/networkGroup/1/ipv4PrefixPools/1*

*AddrStepSupported: True*

*Count: 5*

*DescriptiveName: Basic IPv4 Addresses 1*

*LastNetworkAddress: ['200.1.0.0', '200.2.0.0', '200.3.0.0', '200.4.0.0', '200.5.0.0']*

*Name: Basic IPv4 Addresses 1*

*NetworkAddress: /api/v1/sessions/1/ixnetwork/multivalue/222*

*NumberOfAddresses: 1*

*NumberOfAddressesAsy: /api/v1/sessions/1/ixnetwork/multivalue/225*

*PrefixAddrStep: /api/v1/sessions/1/ixnetwork/multivalue/224*

*PrefixLength: /api/v1/sessions/1/ixnetwork/multivalue/223*

*# To modify the ipv4PrefixPools "networkAddress" value. In Device Group configurations we have*

*a single multiplier so we used a Single value. In-Network Group we have configured Multiplier as 5.*

*So, we will learn to use Increment Values here.*

*prefixObj1.NetworkAddress.Increment(start\_value = '100.1.1.0', step\_value = '1.0.0.0')*

*Create Network Group on topology-2 device group 1:*

*# To create a new network group. We had given the Name of the network group, Multiplier in a single line*

*nwGrpObj2 = dgObj2.NetworkGroup.add(Name="BGP\_NW\_GRP2", Multiplier = 5)*

*# To create IPv4 Prefix Pool for network group*

*prefixObj2 = nwGrpObj2.Ipv4PrefixPools.add()*

*# To modify the ipv4PrefixPools "networkAddress"*

*prefixObj2.NetworkAddress.Increment(start\_value = '110.1.1.0', step\_value = '1.0.0.0')*



Adding BGP network group to device group

## Start Protocols:

*# To start all protocols configured use the session handle object we created initially*

ixnetwork.StartAllProtocols()



Protocol stacks are up

## Configure Traffic:

*Create Traffic Item:*

*# To create new traffic items, use session object with Traffic class*

*trafficItemObj = ixnetwork.Traffic.TrafficItem.add()*

*# To modify the name of the traffic item*

*trafficItemObj.Name = 'BGP topo1 to topo2'*

*# To modify traffic item 1 'srcDestMesh'*

*trafficItemObj.SrcDestMesh = 'one-to-one'*

*# To modify traffic item 1 'allowSelfDestined'*

*trafficItemObj.AllowSelfDestined = False*

*# To modify traffic item 1 'trafficType'*

*trafficItemObj.TrafficType = ‘ipv4’*

*# To modify traffic item 1 'routeMesh'*

*trafficItemObj.RouteMesh = ‘oneToOne’*

*# To modify traffic item 1 'biDirectional’*

*trafficItemObj.BiDirectional = True*

*Create Endpoints:*

*# To create endpoint set for the traffic item use traffic object created above*

*endPointSetObj = trafficItemObj.EndpointSet.add()*

*# To modify the endpoint set ‘sources’ use endpoint object created above and Sources attribute*

*We are creating Traffic with BGP prefixes. So, we are going to use the BGP prefix object created earlier*

*endPointSetObj.Sources = [prefixObj1]*

*# To modify the endpoint set ‘name’*

*endPointSetObj.Name = ‘FlowGroup-1’*

*# To modify the endpoint set ‘destinations’*

*endPointSetObj.Destinations = [prefixObj2]*

*# To modify the traffic item transmissionControl ‘type’ use configElementObj*

*configElementObj = trafficItemObj.ConfigElement.find()*

*configElementObj. TransmissionControl.Type = 'fixedFrameCount'*

*# To modify the traffic item transmissionControl ‘frameCount’ use configElementObj*

*configElementObj. TransmissionControl.FrameCount = 50000*

*# To modify the traffic item frameRate ‘rate’ use configElementObj.FrameRate*

*configElementObj.FrameRate.Rate = 88*

*# To modify the traffic item frameRate ‘type’ use configElementObj.FrameRate*

*configElementObj.FrameRate.Type = 'percentLineRate'*

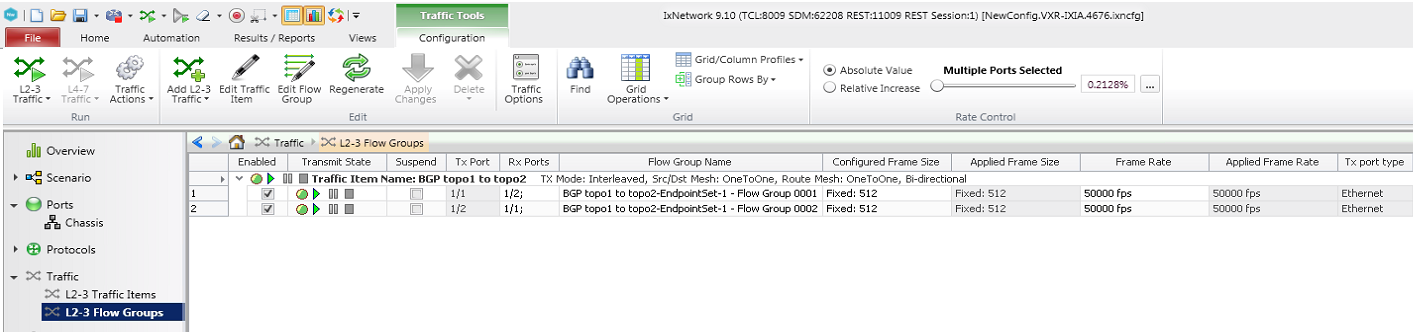
*# To modify the traffic item frameSize ‘fixedSize’ use configElementObj.FrameSize*

*configElementObj.FrameSize.FixedSize=512*

*# To modify the traffic item ‘tracking’ use*

*trackObj = trafficItemObj.Tracking.find()*

*trackObj.TrackBy = ['flowGroup0', 'sourceDestValuePair0']*



Traffic configured with the specified options

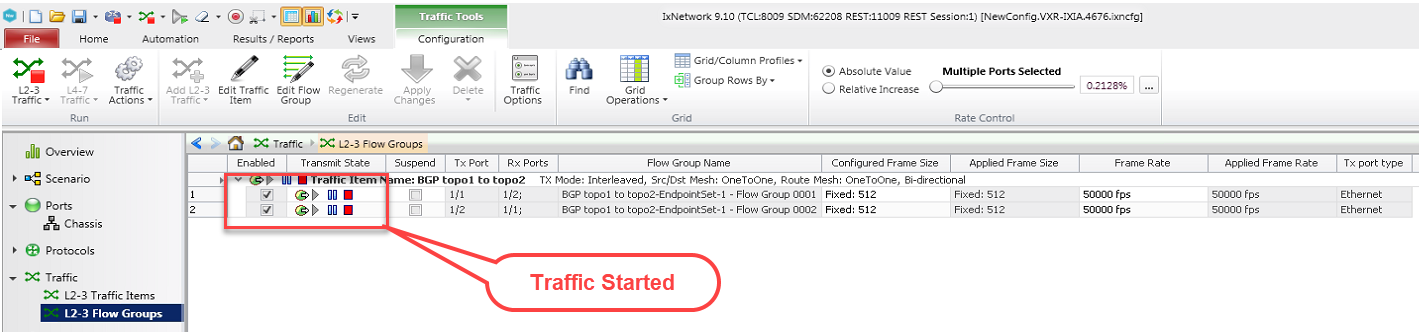
## Start Traffic and Stop Traffic:

*# To apply all configured traffic items, use session object*

*ixnetwork.Traffic.Apply()*

*# To start all configured traffic items*

*ixnetwork.Traffic.Start()*



Traffic Started

*# To stop all the running traffic items*

*ixnetwork.Traffic.Stop()*

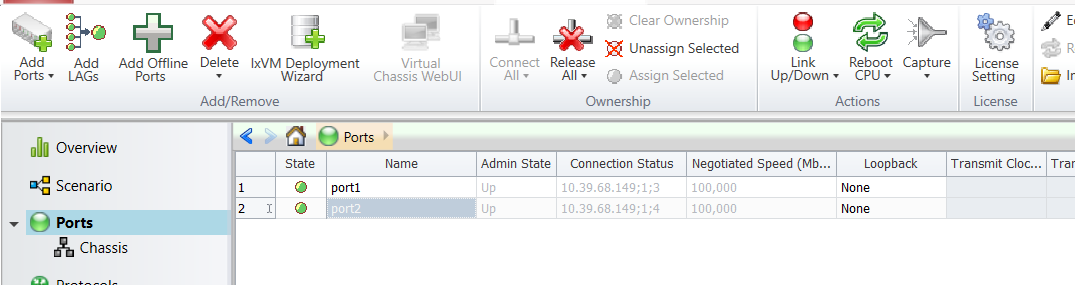
## Disconnect:

*# To delete the IxNetwork session use session object and Remove method*

*ixnetwork.Remove()*

# Using find() to query any Object:

add(), remove() and find() are an important function of RESTPy Library, which can be used in any objects. find() is one of the important functions which used to retrieve any object with Name, ID etc. find() to retrieve vport object:



Port Configured

*To get all vport object:*

As given in the above image, we have two ports connected. We will see how to retrieve the vport object here

*# Use find() to get all vport Object*

*ixnetwork.Vport.find()*

*<ixnetwork\_restpy.testplatform.sessions.ixnetwork.vport.vport.Vport object at 0x03FB7148>*

*# Use find(Name=’port1’) to get the vport object with the name associated with*

*ixnetwork.Vport.find(Name='port1')*

*<ixnetwork\_restpy.testplatform.sessions.ixnetwork.vport.vport.Vport object at 0x03F99B20>*

*# Use find(AssignedTo=’10.39.68.149;1;4’) to get vport object with the physical port associated with*

*vport = ixnetwork.Vport.find(AssignedTo = '10.39.68.149:1:4')*

*print(vport)*

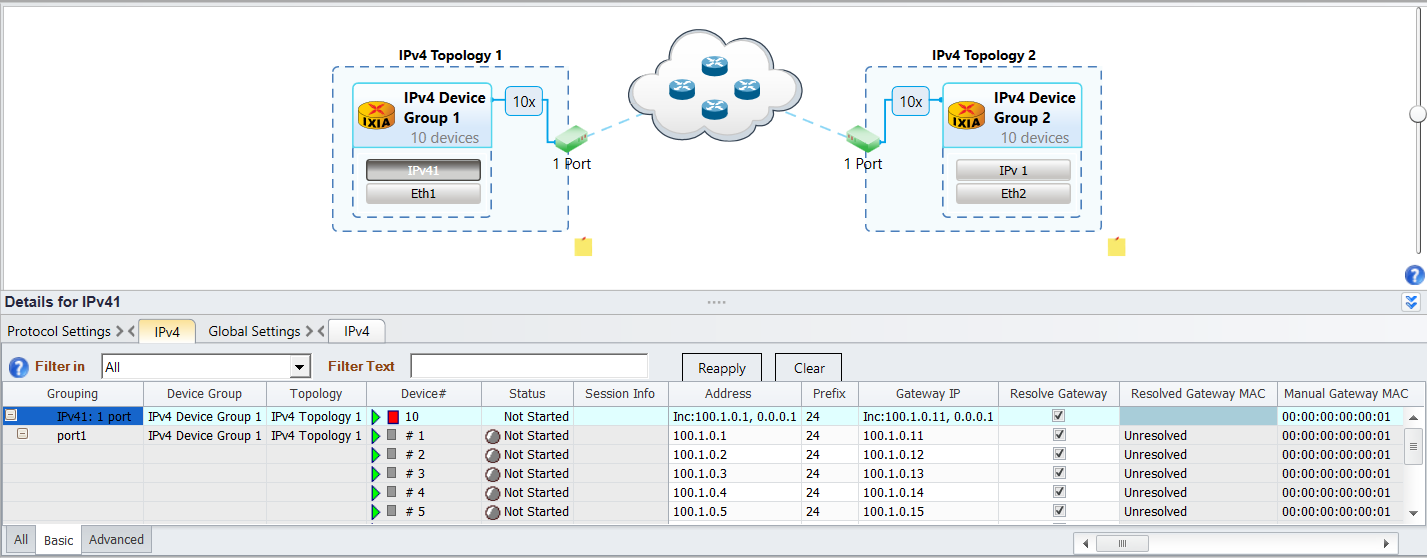
*<ixnetwork\_restpy.testplatform.sessions.ixnetwork.vport.vport.Vport object at 0x037FC3A0>*

*print(vport.AssignedTo)*

*'10.39.68.149:1:4'*

# Configure Multi values through RESTPy:

Multi values are pattern of input value types such as increment, decrement, list and single value. Configuring multi values made very easy using RESTPy. This section provides a walk-through to configure Multi values in the IPv4 stack.



Sample IPv4 Stack

## Get IPv4 Object from configuration using find():

*ipObj1 = ixnetwork.Topology.find(Name='IPv4 Topology 1').DeviceGroup.find(Name='IPv4 Device Group 1').Ethernet.find(Name='Eth1').Ipv4.find(Name='IPv41')*

*print(ipObj1)*

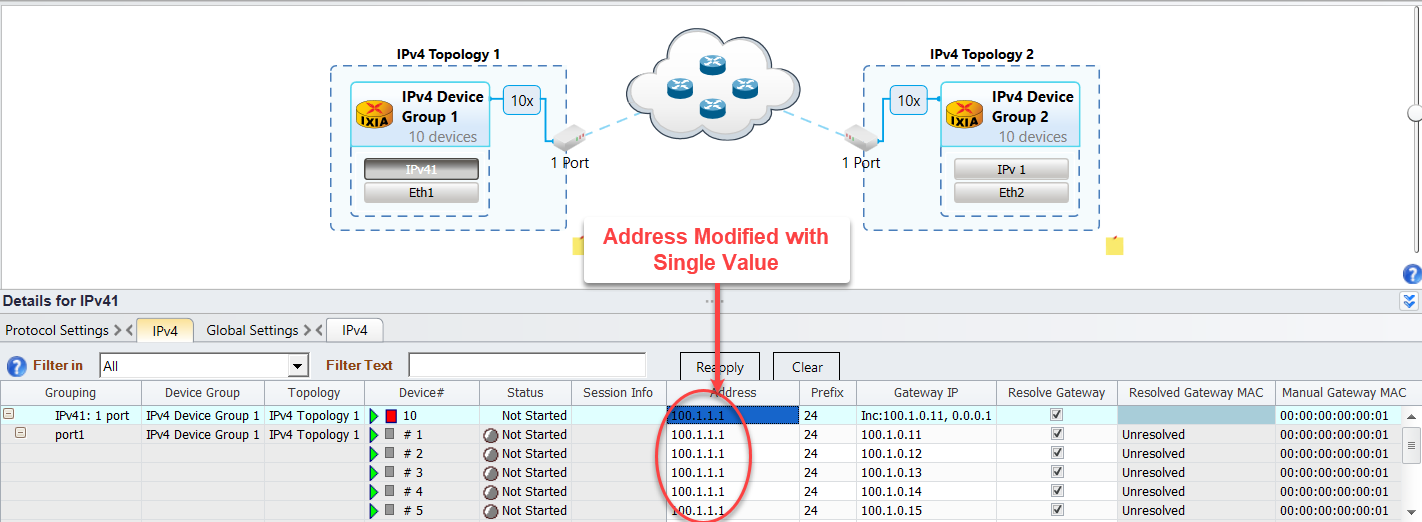
*<ixnetwork\_restpy.testplatform.sessions.ixnetwork.topology.ipv4\_8cb960b62ae85a03e1b40a57bfaeb7bb.Ipv4 object at 0x04113440>*

*print(ipObj1.Address)*

*Inc: 100.1.0.1, 0.0.0.1*

## Modify IPv4 Address with Single Value:

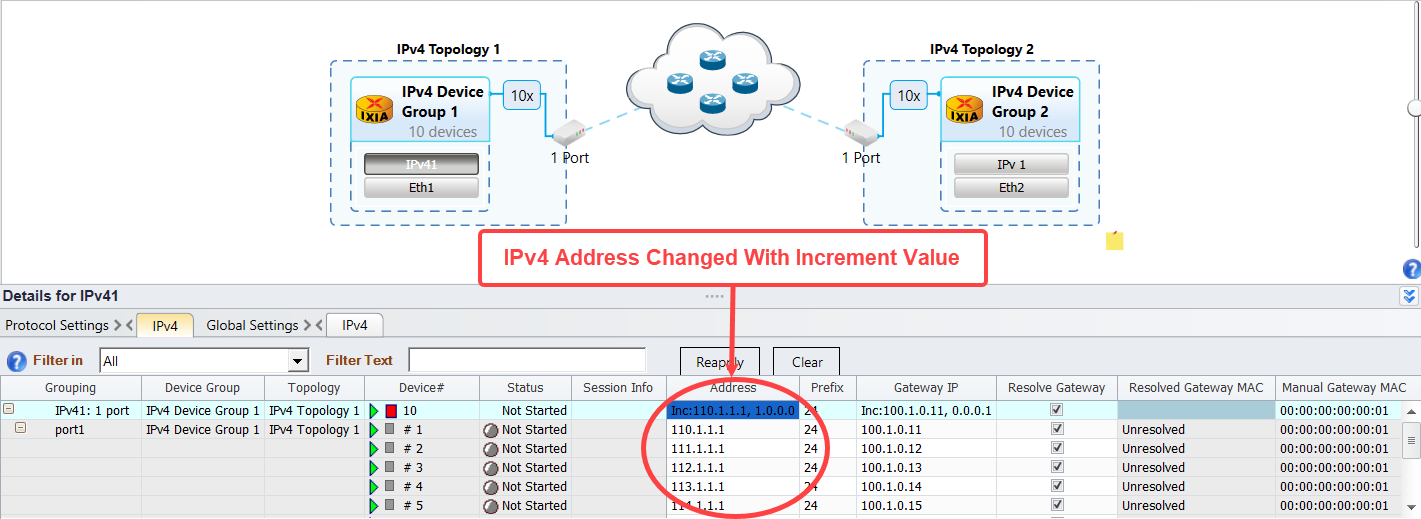
*ipObj1.Address.Single("100.1.1.1")*



IP Address configured with Single Value

## Modify IPv4 Address with Increment Value:

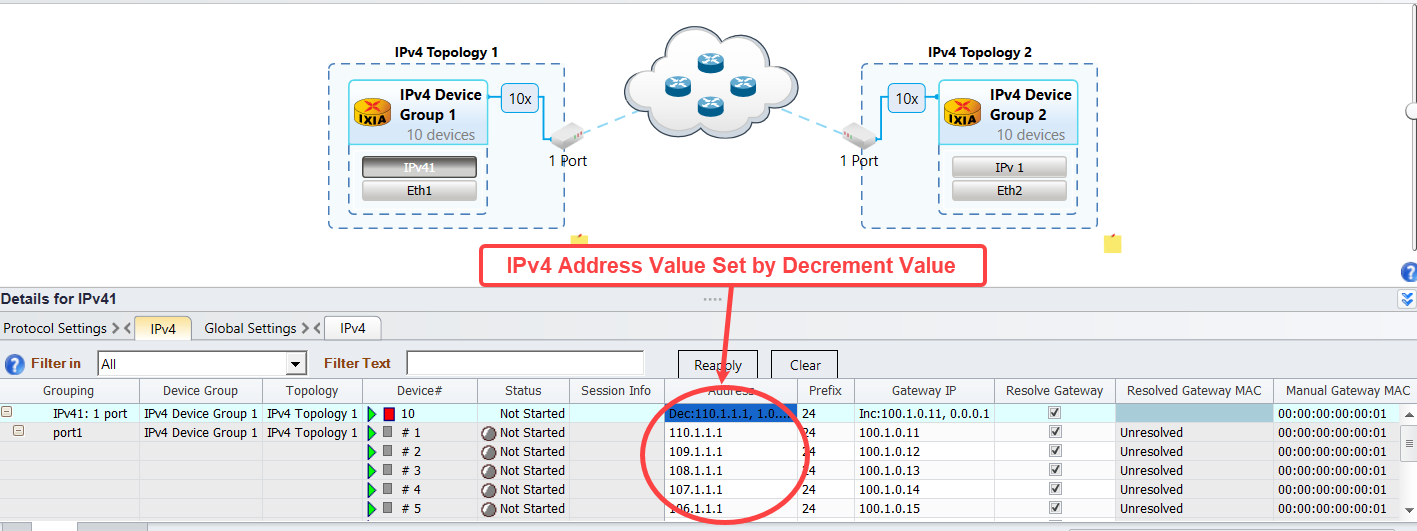
*ipObj1.Address.Increment(start\_value="110.1.1.1",step\_value="1.0.0.0")*



IP Address configured with Increment Value

## Modify IPv4 Address with Decrement Value:

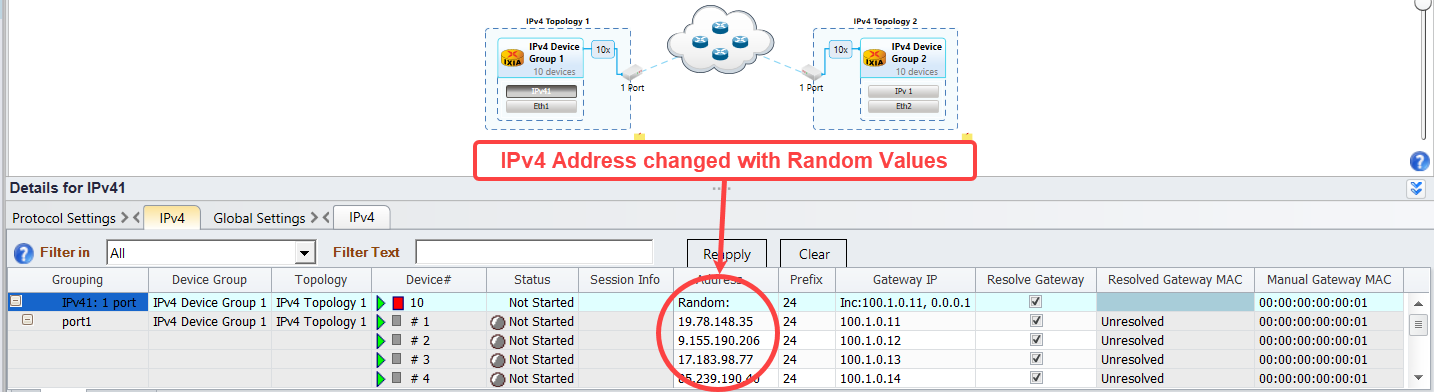
*ipObj1.Address.Decrement(start\_value="110.1.1.1",step\_value="1.0.0.0")*



IP Address configured with Decrement Value

## Modify IPv4 Address with Random Multivalue:

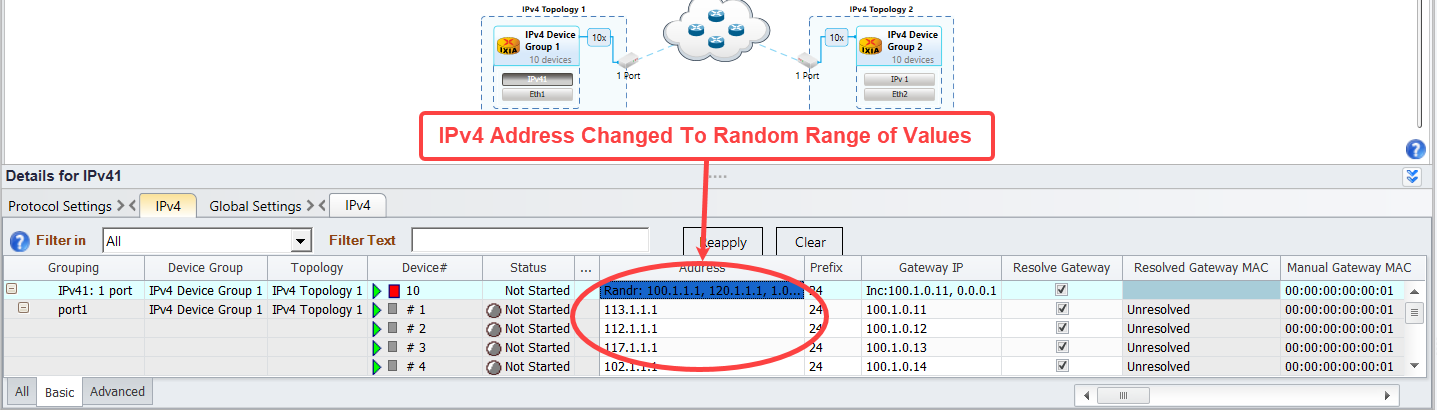
*ipObj1.Address.Random()*



IP Address configured with Random Value

## Modify IPv4 Address with Repeatable Random Range of Multivalue:

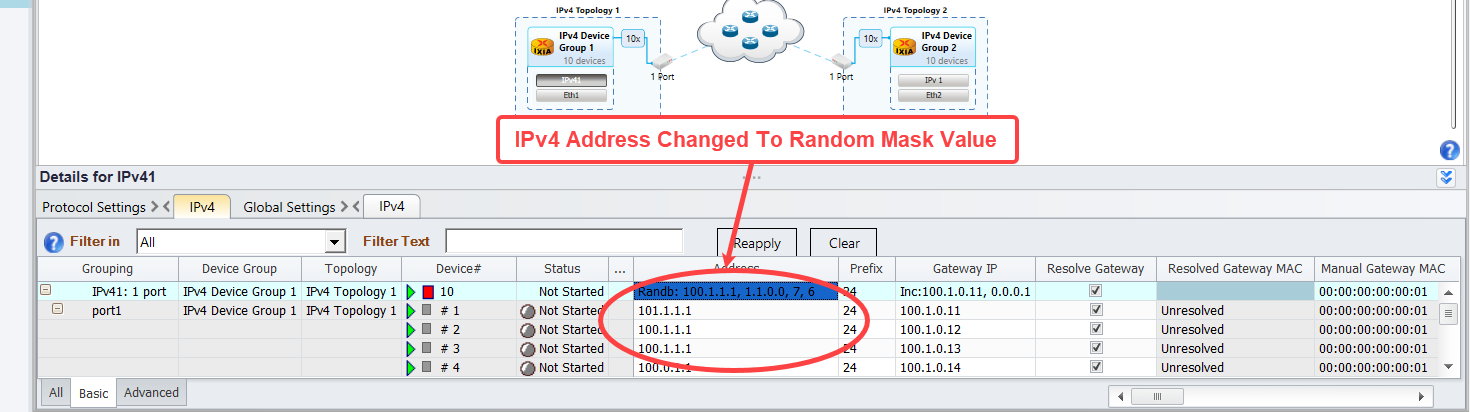
*ipObj1.Address.RandomRange(min\_value='100.1.1.1', max\_value='120.1.1.1', step\_value='1.0.0.0', seed=3)*



IP Address configured with Repeatable Random Range Value

## Modify IPv4 Address with Repeatable Random Mask of Multivalue:

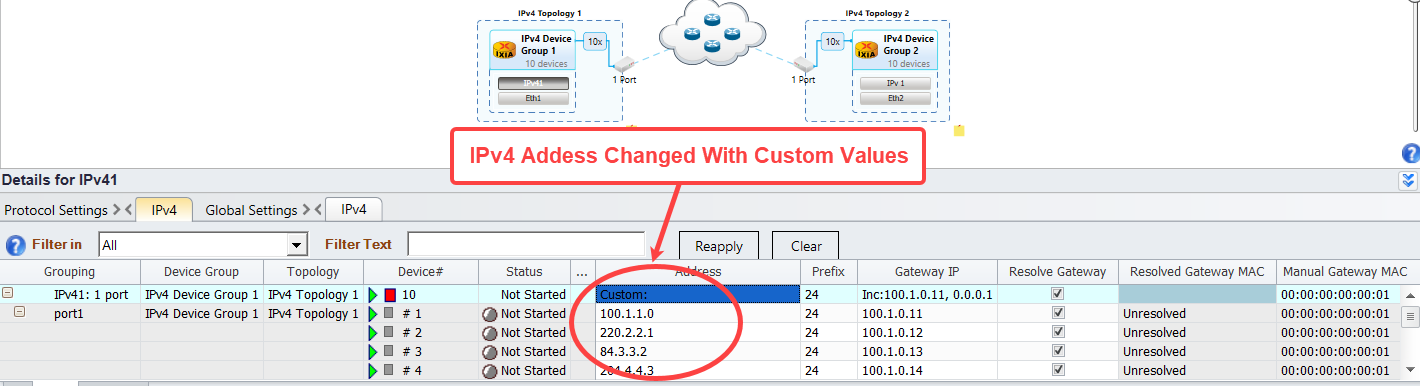
*ipObj1.Address.RandomMask(fixed\_value='100.1.1.1', mask\_value='1.1.0.0', seed=7, count=6)*



IP Address configured with Repeatable Random Mask Value

## Modify IPv4 Address with custom Multivalue:

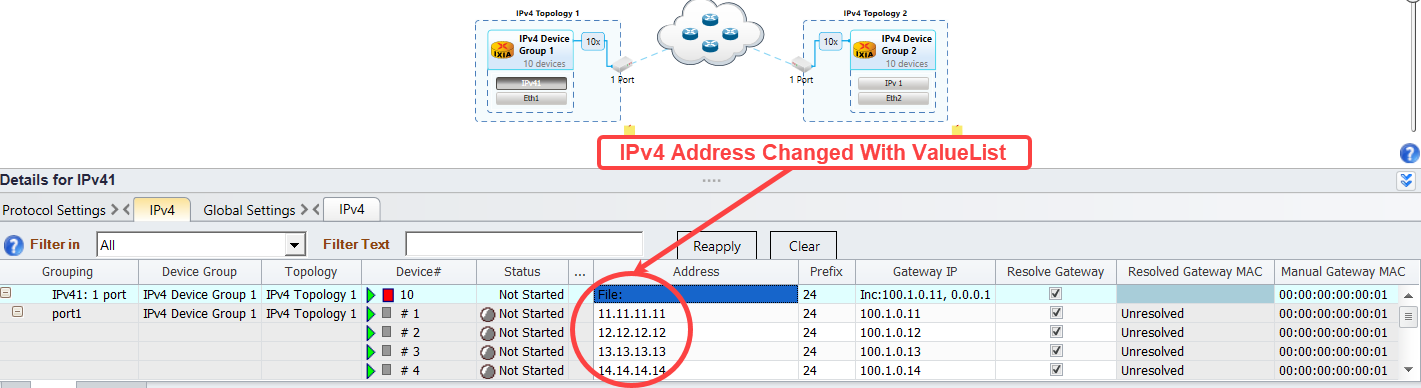
*ipObj1.Address.Custom(start\_value='100.1.1.0', step\_value='1.0.0.0', increments=[('110.1.1.1', 12, [('120.1.1.1', 20, [])])])*



IP Address configured with Custom Value

## Modify IPv4 Address with ValueList Multivalue:

*ipObj1.Address.ValueList(['11.11.11.11','12.12.12.12','13.13.13.13','14.14.14.14','15.15.15.15','16.16.16.16','17.17.17.17','18.18.18.18','19.19.19.19','20.20.20.20'])*



IP Address configured with List of Values

# To Know More on RESTPy:

* [https://openixia.github.io/ixnetwork\_restpy/#/reference](https://openixia.github.io/ixnetwork_restpy/%23/reference)
* [https://openixia.github.io/ixnetwork\_restpy/#/samples](https://openixia.github.io/ixnetwork_restpy/%23/samples)

# Support:

* For more information: <https://support.ixiacom.com/>
* For support assistance, contact: [support.ix@keysight.com](file:///C:\Users\sshimoga\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\6W27DVNM\support.ix@keysight.com)

### Service and Support

|  |  |  |
| --- | --- | --- |
| **Americas** |  |  |
| Brazil  (11) 4197 3600 | Canada  (877) 894 4414 | Mexico  01800 5064 800 |
| United States  (800) 829 4444 |  |  |

|  |  |  |
| --- | --- | --- |
| **Asia Pacific** |  |  |
| Australia  1 800 629 485 | India  1 800 112 929 | Malaysia  1 800 888 848 |
| China  800 810 0189 | Japan  0120 (421) 345 | Singapore  1 800 375 8100 |
| Hong Kong  800 938 693 | Korea  080 769 0800 | Taiwan  0800 047 866 |
| Other Asian Countries: [www.Keysight.com/find/contactus](http://www.agilent.com/find/contactus) |  |  |

|  |  |  |
| --- | --- | --- |
| **Europe & Middle East** |  |  |
| Belgium  32 (0) 2 404 93 40 | Ireland  1890 924 204 | Spain  34 (91) 631 3300 |
| Denmark  45 45 80 12 15 | Israel  972-3-9288-504/544 | Sweden  0200-88 22 55 |
| Finland  358 (0) 10 855 2100 | Italy  39 02 92 60 8484 | Switzerland  0800 80 53 53 |
| France  0825 010 700\* \*0.125 €/minute | Netherlands  31 (0) 20 547 2111 | United Kingdom  44 (0) 118 927 6201 |
| Germany  49 (0) 7031 464 6333 |  |  |
| **Other Unlisted Countries:** [www.Keysight.com/find/contactus](http://www.agilent.com/find/contactus) |  |  |

© Agilent Technologies, Inc. 2014

Printed in Malaysia

April 2014

For Keysight Internal Reference Only  
Manufacturing Part Number  
