NOCTION
NETWORK INTELLIGENCE

Intelligent Routing Platform

Installation and Configuration Guide

[Version 4.2]

1901 S Bascom Ave, Campbell, CA 95008

Tel: 1-650-618-9823

Email: info@noction.com

Contents

Chapter 1

Introduction

1.1 How to contact support

If you encounter any problems while using or setting up the Intelligent Routing Platform, please contact us.

You may contact us in the following ways:

- Email us at support@noction.com
- Open a support request at https://helpdesk.noction.com

1.2 What is IRP

BGP is a fundamental technology for the fault-tolerance of the Internet, which chooses network paths based on the number of hops traffic must traverse before it reaches its destination. However, BGP does not take into account the important factors of network performance. Even if multi-homing does provide some redundancy, multiple network outages have shown that multi-homing alone is not the solution for risk diversity and business continuity. When major blackouts or even congestions happen, multi-homing gives a fallback link for the "first-mile" connection, rather than providing a way to route around the Internet "middle-mile" issues.

Noction Intelligent Routing Platform (IRP) is a product developed by Noction to help businesses to optimize their multi-homed network infrastructure. The platform sits in the network and gets a copy of the traffic from the edge routers. The system passively analyzes it for specific TCP anomalies and actively probes remote destination networks for such metrics as latency, packet loss, throughput, and historical reliability. It computes a performance- or a cost-improvement network traffic engineering policy and applies the new improved route by announcing it to the network's edge routers via traditional BGP session.

Noction IRP is a complete network monitoring and troubleshooting solution, which facilitates the detection, diagnosis, and automatic resolution performance issues. It delivers real-time views and dashboards that allow to visually track network performance and generate triggers, alerts and notifications when specific problems occur.

1.2.1 IRP Features

The Intelligent Routing Platform is designed to help Service Providers to improve the performance and to reduce the costs of running a multi-homed BGP network. The system makes intelligent routing decisions by analyzing various network performance metrics and selecting the best performing route for the traffic to pass through. As a result, Noction IRP allows you to:

- Improve overall network performance
- Reroute congestion and outages

- Decrease network downtime
- Reduce latency and packet loss
- Get comprehensive network performance analytics
- Facilitate network troubleshooting
- Decrease network operational costs
- Monitor platform performance
- Reduce the risk of human errors during BGP configuration

1.2.2 IRP Components

The IRP platform has a few interconnected components (see figure 1.2.1) , which are performing together to improve the routing best path selection and to eliminate various common network issues.

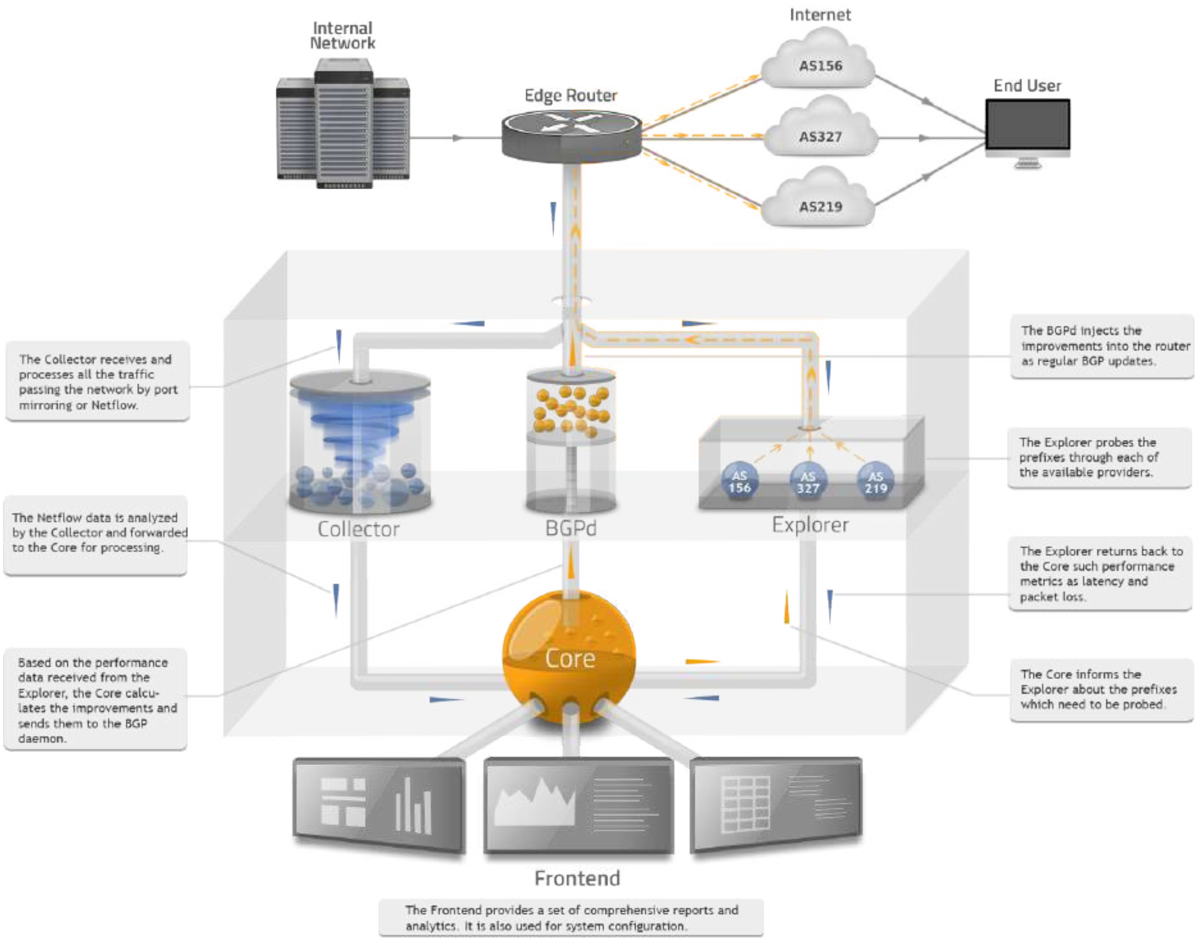

Figure 1.2.1: IRP Components

A short description of each of the components is given below. The detailed information is available in the next chapters.

To inject the Improvements into the network, the platform needs to 'tell' the routers what exactly needs to be changed in the routing table. IRP **BGP daemon** announces the improved prefix with an updated next-hop for the traffic to start flowing through the new path.

Bmpserver receives BMP monitoring sessions to provide route data to IRP components.

Core is the most important part of the system. It runs all the logical operations and interconnects all the components. It handles the performance and cost improvements. It processes, stores and exports data.

Collector **Irpflowd** & **Irpspand** are receiving, analyzing and processing all the traffic passing the network. They have two ways to gather data about the network: by mirroring network traffic or by using NetFlow/sFlow. Collector also gathers interface statistics from the edge routers via SNMP protocol.

Explorer runs all the probes and checks the metrics specified by the platform policies, such as packet loss and latency. This information is sent back to the Core.

Frontend represents a web interface with a comprehensive set of reports, graphs and diagnostic information which can reflect the current and historical network state, as well as the benefits of a single or multiple Intelligent Routing Platforms in terms of network optimization.

Globalcc performs communication between different IRP instances to maintain commit levels globally.

Irpapid is used to serve requests from Frontend (the Global Management Interface) and provides API to external tools.

Irpdetectd collects various statistics for DDOS detection and mitigation feature.

Irpinperfd performs statistical analyzis for Inbound Performance.

Irpmng is a command-line interface designed to perform various management tasks.

Irppushd is responsible for forwarding events to configured notification channels.

1.2.3 IRP Technical Requirements

In order to plan the IRP deployment in your network, a series of requirements need to be met and specific information has to be determined to configure IRP.

1.2.3.1 Hardware requirements

⚠ In production, a dedicated server for each IRP instance is strongly recommended. The system can also be deployed on a Virtual Machine with matching specifications, provided that this is hardware- or paravirtualization (Xen, KVM, VMware). Os-level virtualization (OpenVZ/Virtuozzo or similar) is not supported.

1. CPU
   - Recommended Intel® Xeon® Processor E3/E5 family, for example:
     - 1x Intel® Xeon® Processor E3 family for up to 20 Gbps traffic;
     - 1x Intel® Xeon® Processor E5 family for 40 Gbps or more traffic.
2. RAM
   - if providing sFlow/NetFlow data at least 16 GB, recommended - 32 GB;
   - if providing raw traffic data by port mirroring:
     - minimum 16 GB for up to 10 Gbps traffic;
     - minimum 32 GB for 40 Gbps traffic
   - Additional RAM would be required to maintain large amount of BGP & BMP sessions (for example, Bgpd occupies about 10G of RAM for 16 full view BGP sessions; the estimation may change due to growth of world's BGP table and new IRP features).
3. HDD
   - At least 160GB of storage;
   - SAS disks are recommended (SSDs are required only for 40Gbps+ networks);
   - HDD partitioning:
     - LVM is recommended;
     - At least 100GB disk space usable for /var or separate partition;

- At least 10GB disk space usable for /tmp or separate partition. This is required for big mysql tables manipulation. More disk space might be required under heavy workload.

4. NIC

- if providing sFlow/NetFlow data - at least 1 x 1000Mbps NIC while two NICs are recommended (one will be dedicated to management purposes).
- if providing raw traffic data by port mirroring - additional 10G interfaces are required for each of the configured SPAN ports (Myricom 10G network cards with Sniffer10G license are recommended to be used for high pps networks). When configuring multiple SPAN ports the same number of additional CPU cores are needed to analyze traffic.

⚠ In very large networks carrying hundreds or more of Gbps of traffic and in IRP configurations with very aggressive optimization settings configurations with 2x or 4x CPUs are recommended. The IRP servers in these cases should also allocate double the recommended RAM and use SSD storage.

Noction can size, setup and mail to you an appliance conforming to your needs. The appliance is delivered with OS installed and IRP software deployed.

ℹ A different supported OS can be installed on customer request.

1.2.3.2 Software requirements

Clean Linux system, with the latest Red Hat Enterprise Linux 8/9 (or binary alternatives such as RockyLinux) or Ubuntu Server LTS for x86_64 architecture installed on the server.

⚠ IRP has a dependency on MySQL/MariaDB server and it expects the latest version from official OS repositories. In case the DBMS has been installed from a different repository it is strongly advised that the database instance and its configuration is purged before proceeding with IRP installation.

IRP requires root access to local database instance during first installation. In case the root access can't be given, use the statements below to grant all necessary privileges to the 'irp' user and database.:

GRANT ALL ON $dbdbname.* TO '$dbusername'@'$dbourhost' IDENTIFIED BY '$dbpassword' WITH GRANT OPTION

GRANT SELECT ON $dbdbname_fe.* TO '$dbusername_fe'@'$dbourhost' IDENTIFIED BY '$dbpassword_fe'

where $dbdbname (4.1.7), $dbusername (4.1.12), $dbpassword (4.1.10), $dbourhost (4.1.9) are the corresponding parameters from `/etc/noction/db.global.conf`

1.2.3.3 Network-related information and configuration

IRP is designed to help Service Providers (AS) optimize a multi-homed BGP network. This implies the basic prerequisites for using IRP:

- Ownership of the AS for the network where IRP is deployed,
- BGP protocol is used for routing and,
- Network is multi-homed.

Eventually the following needs to be performed in order to deploy and configure IRP:

1. Prepare a network diagram with all the horizontal (own) as well as upstream (providers) and downstream (customers) routers included. Compare if your network topology is logically similar to one or more of the samples listed in section Collector Configuration for example Flow export configuration .

2. Identify the list of prefixes announced by your AS that must be analyzed and optimized by IRP.

3. Review the output of commands below (or similar) from all Edge Routers:

   - ‘**sh ip bgp summary**‘
   - ‘**sh ip bgp neighbor [neighboor-address] received-routes**‘
   - ‘**sh ru**‘ (or similar)

   The settings relating to BGP configuration, prefixes announced by your ASN, the route maps, routing policies, access control list, sFlow/NetFlow and related interfaces configurations are used to setup similar IRP settings or to determine what settings do not conflict with existing network policies.

4. Provide traffic data by:

   (a) **sFlow**, **NetFlow** (v1, 5, 9) or **jFlow** and send it to the main server IP. Make sure the IRP server gets both inbound and outbound traffic info.
   Egress flow accounting should be enabled on the provider links, or, if this is not technically possible, ingress flow accounting should be enabled on all the interfaces facing the internal network.
   NetFlow is most suitable for high traffic volumes, or in the case of a sophisticated network infrastructure, where port mirroring is not technically possible.
   Recommended sampling rates:

      i. For traffic up to 1Gbps: 1024
      ii. For traffic up to 10Gbps: 2048

   (b) Or: configure port mirroring (a partial traffic copy will suffice). In this case, additional network interfaces on the server will be required - one for each mirrored port.

   See also: Collector Configuration

5. Setup Policy Based Routing **(PBR)** for IRP active probing.

   - Apart from the main server IP, please add an additional alias IP for each provider and configure PBR for traffic originating from each of these IPs to be routed over different providers.
   - No route maps should be enforced for the main server IP, traffic originating from it should pass the routers using the default routes.
   - Define Provider↔PBR IP routing map

   In specific complex scenarios, traffic from the IRP server should pass multiple routers before getting to the provider. If a separate probing Vlan cannot be configured across all routers, GRE tunnels from IRP to the Edge routers should be configured. The tunnels are mainly used to prevent additional overhead from route maps configured on the whole IRP⟷Edge routers path.

> If network has Flowspec capabilities then alternatively Flowspec policies can be used instead of PBR. Refer for example Flowspec policies, global.flowspec.pbr.

1. Configure and provide **SNMP** for each provider link, and provide the following information:

   - SNMP interface name (or ifIndex)
   - SNMP IP (usually the router IP)
   - SNMP community

This information is required for the report generation, Commit Control decision-making and prevention of overloading a specific provider with an excessive number of improvements.

The above is applicable in case of SNMP v2c. If SNMP v3 is used further details will be required depending on security services used.

1. To setup cost related settings as well as the Commit Control mechanism, provide the maximum allowed interface throughput for each provider link as well as the cost per Mbps for each provider.

1.2.4 IRP Operating modes

The IRP platform can operate in two modes, which can be used at different stages of the deployment process. During the initial installation and configuration, it is recommended for the system not to inject any improvements into the network until the configuration is completed.

After running several route propagation tests, the system can be switched to the full Intrusive mode.

1.2.4.1 Non-intrusive mode

While running in this mode, the system will not actually advertise any improvement to the network, and will only reflect the network improvements and events in the platform reports and graphs.

1.2.4.2 Intrusive mode

After the system configuration is completed, and manual route propagation tests were performed in order to ensure that the edge routers behavior is correct, the system can be switched to Intrusive mode. While running in this mode, the system injects all the computed improvements into the edge router(s) routing tables, allowing the traffic to flow through the best performing route.

1.2.4.3 Going Intrusive

While IRP operates in non-intrusive mode it highlights the potential improvements within client's environment. Going Intrusive will realize IRP potential.

The difference between Intrusive and Non-Intrusive operating modes is that IRP advertises improvements to edge routers. In order to switch to Intrusive we follow a controlled process. The highlights of the process are as follows:

1. The optimizing component of IRP (Core) is taken offline and existing improvements are purged. The Core being offline guarantees IRP will not automatically insert new improvements into its Current Improvement table and hinder the Go Intrusive process.

Listing 1.1: Stop IRP Core and purge existing improvements

```
root@server ~ $ systemctl stop core
root@server ~ $ mysql irp -e 'delete␣from␣improvements;'
```

2. Enable Intrusive Mode and adjust IRP Core and Bgpd parameters as follows:

Listing 1.2: Switch to Intrusive Mode and adjust IRP Core and Bgpd parameters

```
root@server ~ $ nano /etc/noction/irp.conf
global.nonintrusive_bgp = 0
core.improvements.max = 100
bgpd.improvements.remove.next_hop_eq = 0
bgpd.improvements.remove.withdraw = 0
```

3. Improvements towards test networks are introduced manually so that client's traffic is not affected. The improvements are chosen so that they cover all client's providers. Any public networks could be used for test purposes. Just keep in mind that preferably your network shouldn't have traffic with chosen test network in order to do not re-route the real traffic. Use the template below in order to insert the test improvements:

   Listing 1.3: Inserting test improvements

```
mysql> insert into improvements
  (ni_bgp, prefix, peer_new, ipv6, asn)
  values
    (0, '10.10.10.0/24', 1, 0, 48232),
    (0, '10.10.11.0/24', 2, 0, 48232),
    (0, '10.10.12.0/24', 3, 0, 48232);
```

4. Make sure that 'route-reflector-client' is set for IRP BGP session.

5. Make sure that 'next-hop-self' is not configured for IRP BGP session.

6. On iBGP sessions (between edge routers, route-reflectors; except session with IRP) where 'next-hop-self' is configured, the following route-map should be applied:

   Listing 1.4: Remove next-hop-self route-map (RM-NHS) example

```
route-map RM-NHS
set ip next-hop peer-address
neighbor X.X.X.X route-map out RM-NHS
```

   where X.X.X.X is the iBGP neighbor

> ⛔ route-map contents should be integrated into existing route-map in case other route-map already configured on the iBGP session.

7. Use the commands below to restart IRP Bgpd to use actual IRP configuration and establish BGP session(s) and verify if BGP updates are being announced:

   Listing 1.5: Restart IRP Bgpd

```
root@server ~ $ systemctl restart bgpd
root@server ~ $ tail -f /var/log/irp/bgpd.conf

Wait for the following lines for each BGP session:
NOTICE: Adding peer X
NOTICE: BGP session established X
INFO: N update(s) were sent to peer X
```

   where X is the router name and N is the number of the updates sent towards the X router.

8. Verify if IRP BGP announcements are properly propagated across all the network. Run the following commands on each router (the commands vary depends on the router brand):

   Listing 1.6: Show BGP information for specified IP address or prefix

```
show ip bgp 10.10.10.1
show ip bgp 10.10.11.1
show ip bgp 10.10.12.1
```

Analyze the output from all the routers. If the IRP BGP announcements are properly propagated, you should see /25 (refer to 4.4.37) announcements and the next-hop for each announcement should be the improved provider's next-hop:
10.10.10.1 - provider 1 next-hop
10.10.11.1 - provider 2 next-hop
10.10.12.1 - provider 3 next-hop
(refer to 4.14.31, 4.14.39, 4.17.3).

Run the following commands in order to check if IRP improvements are announced and applied:

Listing 1.7: Traceroute destination networks

```
root@server ~ $ traceroute -nn 10.10.10.1
root@server ~ $ traceroute -nn 10.10.11.1
root@server ~ $ traceroute -nn 10.10.12.1
```

Again, you should see corresponding providers' next-hops in the traces.

9. If the tests are successful perform the steps below:

   (a) Delete test improvements

   Listing 1.8: Delete test improvements

   ```
   root@server ~ $ mysql -e "delete from improvements where prefix 
      like '10.10.1%';"
   ```

   (b) Configure at most 100 improvements and revert Bgpd configuration

   Listing 1.9: Configure the maximum improvements limit and revert Bgpd configuration

   ```
   root@server ~ $ nano /etc/noction/irp.conf
   core.improvements.max = 100
   bgpd.improvements.remove.next_hop_eq = 1
   bgpd.improvements.remove.withdraw = 1
   ```

   (c) Restart IRP Core and Bgpd

   Listing 1.10: Restart IRP Core and Bgpd

   ```
   root@server ~ $ systemctl restart bgpd core
   ```

10. If everything goes well, after 1-2 hours the maximum number of improvements announced is increased to 1000 and after 24 hour to 10000.

As a rollback plan we have to revert the changes and switch the system to non-intrusive mode:

1. Delete test improvements

Listing 1.11: Delete test improvements

```
root@server ~ $ mysql -e "delete from improvements where prefix like 
   '10.10.1%';"
```

2. Switch the system to non-intrusive mode

Listing 1.12: Switch the system to non-intrusive mode

```
root@server ~ $ nano /etc/noction/irp.conf
global.nonintrusive_bgp = 1
```

3. Restart IRP Core and Bgpd

Listing 1.13: Restart IRP Core and Bgpd

```
root@server ~ $ systemctl restart bgpd core
```

1.2.5 BGP Monitoring

IRP uses two types of BGP monitors and a BMP monitoring station to collect data, diagnose and report mainly the state of the BGP session between the edge routers and the providers, as well as the network reachability through a specific provider. The information provided by monitors enables IRP to avoid announcing routing updates that would result in traffic misrouting for example by sending improvements to a failed provider but also to better inform IRP probing and improvement decisions.

1.2.5.1 Internal monitor

Internal BGP Monitor is checking the state of the Edge Router→Provider BGP session by regularly polling the router via SNMP. When queried, the SNMP protocol returns variables describing the session status to be used by the IRP's Internal BGP Monitor. If the session between the edge router and the provider is down, SNMP will return a value, representing session failure and IRP will react as follows:

- the provider will be marked as FAILED,
- all the improvements towards this provider will be withdrawn from the routing tables to avoid creating black holes,
- new improvements towards this providers will not be made.

In some cases (e.g. DDoS attack or various factors causing router CPU over-usage) there may be no response to the SNMP queries at all. In this case a timeout status will be reported to the Internal Monitor and a 30 minutes timer (called longhold timer) (bgpd.mon.longholdtime) will be started. During this time the monitor will be sending ICMP/UDP ping requests toward the configured provider's next-hop IP address (peer.X.ipv4.next_hop or peer.X.ipv6.next_hop). The requests will be sent once in keepalive period (a parameter adjustable in the BGP daemon configuration interface) (bgpd.mon.keepalive). If the next-hop stops responding to these requests, another 30 seconds timer (called hold timer) (bgpd.mon.holdtime) will be started. If according to the ping response the session is reestablished during this time, the hold timer will be discarded while the longhold timer continues. In case one of the timers expires, the provider is switched to a FAIL state and all the improvements towards this provider will be withdrawn from the routing table. However, if the BGP session with the provider is re-established, the system will start rerouting traffic to this provider.

When the Bgpd is started, the monitors are initialized and one SNMP query is sent towards each router, in order to check the status of the BGP sessions with providers. If there is no reply, the Internal Monitor will send up to two more SNMP requests, separated by a keepalive interval.

Internal monitors for Internet Exchange peering partners are not initialized until there is an improvement made towards it. When running in non-intrusive mode internal monitors for IX peers are not initialized at all.

If none of the SNMP queries returned a status confirming that the sessions with providers are up, the provider will be assigned a FAIL status and the Internal Monitor will continue the periodical SNMP polling (each 60 seconds), to recheck providers sessions' status.

Then, the BGP session with the edge routers is initialized and Bgpd starts retrieving the routing table from the edge routers. While IRP retrieves the routing table, SNMP request may timeout due to the high CPU usage on the edge routers.

For details refer:
bgpd.mon.guardtime
bgpd.mon.keepalive
bgpd.mon.holdtime
bgpd.mon.longholdtime

1.2.5.2 External monitor

External BGP Monitor analyzes the network reachability through a specific provider. It performs ICMP/UDP ping requests towards the configured remote IP address(es) (peer.X.ipv4.mon or peer.X.ipv6.mon) through the monitored provider. If any of the configured IP addresses are accessible, the monitor is marked as OK. If the monitored remote IP addresses do not reply through the examined provider IRP will react as follows:

- the provider will be marked as FAILED,
- all the improvements towards this provider will be withdrawn from the routing table,
- new improvements towards this providers will not be made.

If for some reason (e.g. when the provider's interface goes down state), the Next-Hop of the Policy Based Routing rule does not exist in the routing table, then the packets forwarding may return to the default route. In that case, the External BGP Monitor will return a false-positive state. To avoid that by properly configuring PBR, please consult "Specific PBR configuration scenarios" (Specific PBR configuration scenarios).

The External BGP Monitor status does not depend on the state of the BGP session(s) between the edge router and the provider (which is monitored by the Internal BGP Monitor). Therefore, in the case that the BGP session with one of the providers goes down, the External Monitor still shows an OK state which will remain unchanged as long as the packets are successfully routed towards the monitored destination.

We do recommend adding at least two remote IP addresses, in order to prevent false-positive alerts.

When both BGP monitors are enabled (peer.X.mon.ipv4.internal.state, peer.X.mon.ipv4.external.state), they function in conjunction with each other. If any of them fails, the provider will be declared as FAILED and IRP will react as described above. The BGP monitors' statuses are displayed on the system dashboard as shown in the screenshot below.

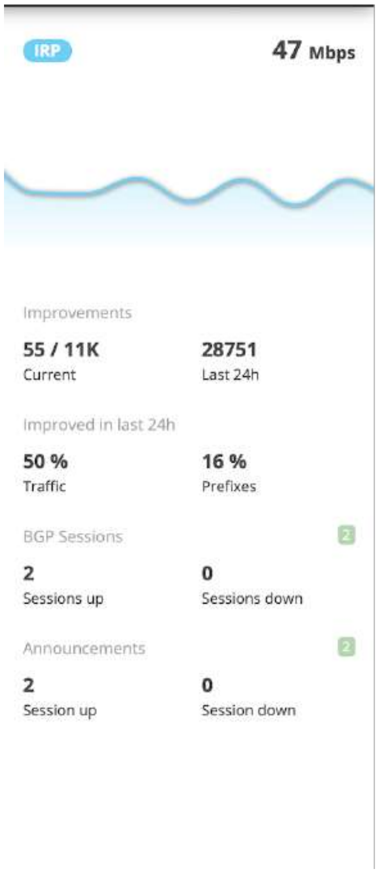

Figure 1.2.2: System Dashboard

Starting with version 1.8.5, IRP requires at least the Internal Monitor to be configured. Otherwise, the system Frontend will return an error as shown below.

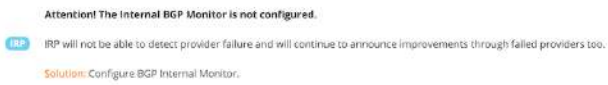

Figure 1.2.3: Error: Internal BGP Monitor not configured

For details refer:
peer.X.mon.snmp
peer.X.ipv4.mon
peer.X.ipv6.mon
peer.X.mon.ipv4.bgp_peer
peer.X.mon.ipv6.bgp_peer

1.2.5.3 BMP monitoring station

A BMP monitoring station is included in IRP starting with version 3.9. It implements the monitoring station specified in RFC 7854 BGP Monitoring Protocol (BMP). The BMP monitoring station requires a monitored router to communicate over BMP the detailed routing information received from neighbors.

The BMP monitoring station exposes detailed routing data to other IRP components so that better and timelier decisions are made, for example:

- BMP lists both active and inactive routes advertised by peers on an Internet Exchange. The additional information is used by IRP to evaluate and identify the best candidate peers at all times. Without BMP data IRP has knowledge about active routes only which only point to a single peer on the IX while all the alternatives are hidden.

- route changes even for inactive routes are visible via BMP. This allows IRP the opportunity to revisit previously made probes and improvements not only at predefined re-probing intervals but also when route changes are detected for both active and inactive routes.
- prefix monitors for IX improvements consume significant router CPU resources in order to service the SNMP requests traversing the router's relevant OIDs. More so this information is at times inaccurate and vendor dependent. When BMP data is available IRP uses this routing data to determine if IX peers still advertise the routes and no longer makes the SNMP requests for those prefixes thus significantly reducing the CPU overhead especially on routers servicing very large IX.
- IRP reconstructs the AS Path for candidate providers in order to make accurate iBGP announcements of improvements. Unfortunately network configuration practices might cause some errors during reconstruction of AS Paths using traceroute. BMP data makes the reconstruction of AS Path redundant and more accurate as this BGP attribute can be retrieved from actual (inactive) routes received from neighbors.
- improvements can be re-visited on AS Path changes. Both new and old provider AS Path attributes are monitored via BMP for changes. When changes are detected IRP re-probes the prefix to ensure the network uses the best available route. Note that re-probing can be triggered on any AS Path changes or only on major ones - when AS Path traverses a different set of autonomous systems.
- Accurate as_path reconstruction helps Inbound performance feature detecting path changes and correlate it with performance changes.

The possible benefits of passing BMP data to IRP are many. To benefit from them the monitored router must support BMP too. Configuration is fully performed on monitored router by pointing it to the IRP BMP monitoring station IP address and port. The monitored router establishes the TCP connection and communicates the data while the IRP BMP monitoring station continuously listens and accepts fresh routing data as it comes.

As per BMP RFC requirements IRP BMP monitoring station never attempts to establish BMP or any other connections with the monitored router leaving the full scope of decisions regarding when and if BMP data is communicated in network's responsibility.

Any route filtering applied to a BGP session with Partial Routing provider or IX peering partner wouldn't not be taken into consideration by a BMP sender. If filtering is deemed as important, then IRP shouldn't use BMP data to find routes (peer.X.bmp.check_routes).

1.2.6 Outage detection

A complete traffic path from source to destination typically passes through multiple networks, with different AS numbers. This is reflected in the traceroute results. If the Outage detection is enabled in the IRP configuration, the system gathers network performance information for all traceroute hops over which the traffic passes to the remote networks. Next, each hop is translated into AS numbers. In case any network anomalies are detected on a specific ASN, then this ASN and the immediate neighbor ASN are declared a problematic AS-pattern. The system then re-probes the prefixes that pass through this AS-pattern. In case the issue is confirmed, all related prefixes are rerouted to the best performing alternate provider.

The Outage detection uses a statistical algorithm for selecting the best routing path. Rerouting will occur for all the prefixes that are routed through the affected as-pattern, despite their current route .

Several improvements-related reports need to indicate the original route for a specific prefix. This value is taken from the last probing results for this prefix, even if the results are outdated (but not older than 24h). Since the outage-affected prefixes are rerouted in bulk by as-pattern, in some cases the reports can show the same provider for both the old and the new route.

1.2.7 VIP Improvements

The VIP Improvements is a feature that allows manual specification of a list of prefixes or AS numbers that will be periodically probed by IRP and optimized in compliance with the probing results. This allows the system to monitor specific networks or Autonomous Systems, without reference to the data provided by the IRP collector.

Possible usage scenarios include, but are not limited to:

- monitoring and optimizing traffic to commercial partners that should have access to your networks via the best performing routes
- monitoring and optimizing traffic to your remote locations, operating as separate networks
- monitoring and optimizing traffic to AS, which are known for the frequent accessibility issues due to geographical or technical reasons

If a prefix is being announced from multiple Autonomous Systems, you can see different ASNs in VIP Improvements report in the prefixes translated from ASN

IRP performs the proactive (more frequent than regular probing) monitoring of the VIP prefixes/ASNs that allows VIPs to be constantly improved.

For future reference, see:
core.vip.interval.probe

1.2.8 Retry Probing

Retry Probing is a feature that allows reconfirmation of the initial and already reconfirmed improvements validity. The feature is applicable to all types of improvements made by the system (Performance, Cost and Commit Control improvements). The improvements that were made more than a retry probing period ago (core.improvements.ttl.retry_probe) are being sent to retry probing. If the probing results confirm that the current improvement is still valid, it stays in the system and its description is updated. Otherwise, it will be removed with the log message further described in this section.
During Retry Probing reconfirmation the improvement details will be updated in the following cases:

- Performance and Cost improvements
  - An old provider has been removed from the system configuration.
    Example: “Old provider and performance metrics not known. New packet loss 55%, avg rtt 105 ms.”
- Commit Control improvements
  - An old provider’s has been removed from the system configuration.
    Example: “Previous provider not known. Rerouted 1 Mbps to Peer5[5] (250 Mbps, 50%)”
  - An old provider’s bandwidth statistics are not available.
    Example: “Rerouted 6 Mbps from Peer1[1] to Peer5[5] (250 Mbps, 50%)”
  - A new provider’s bandwidth statistics are not available.
    Example: “Rerouted 6 Mbps from Peer1[1] (250 Mbps, 50%) to Peer5[5]”
  - The old and new providers’ bandwidth statistics are not available.
    Example: “Rerouted 6 Mbps from Peer1[1] to Peer5[5]”

Commit control improvements are reconfirmed based on their average bandwidth usage (and not on current bandwidth usage). This way if performance characteristics allow it, even when current bandwidth usage is low but the average is still relevant, the improvement is preserved thus anticipating network usage cycles and reducing number of route changes.

During Retry Probing reconfirmation the improvements will be removed from the system and the details will be logged into the log file (core.log) in the following cases:

- The Commit Control feature has been disabled.
  Example: "Prefix 1.0.2.0/24 withdrawn from Improvements (Commit Control is disabled)"
- The low prefix traffic volume is less than the configured bandwidth limits (core.commit_control.agg_bw_min).
  Example: "Prefix 1.0.2.0/24 withdrawn from Improvements (low traffic volume, irrelevant for the Commit Control algorithm)"
- The system has been switched from the Cost mode to the Performance mode (applied for cost improvements only).
  Example: "Prefix 1.0.2.0/24 withdrawn from Improvements (Performance Improvements mode)"
- A prefix has been added to the ignored networks/ASN list.
  Example: "Prefix 1.0.2.0/24 withdrawn from Improvements (added to ignored networks/ASN)"
- The improvement's performance metrics are not the best ones anymore.
  Example: "Prefix 1.0.2.0/24 withdrawn from Improvements (Performance Improvement not actual anymore)"
- The maximum number of improvements limits (core.improvements.max, core.improvements.max_ipv6) are exceeded
  Example: "Prefix 1.0.2.0/24 withdrawn from Improvements (no more available Improvement slots)"

For future references, see:
core.eventqueuelimit.retry_probe_pct
core.improvements.ttl.retry_probe

1.2.9 Routing Policies

Routing Policies brings the capability of defining specific routing policies according to business objectives. This feature permits denying or allowing providers to be used for probing and reaching a specific prefix or ASN. It also provides the possibility to set a static or static exact route through a particular provider. Static route policy will only apply improvements for prefixes learned from the BGP table or external sources. Static exact route policy will announce improvements according to the exact policy details provided by the user, regardless of prefixes being present in the BGP table or external data sources (BGP split does not apply to static exact policies).

Within an Allow or Deny policy, you can choose between VIP or non-VIP (regular) probing mechanisms to be used.

Policies can be configured for networks (ASN) or aggregate prefixes. IRP works with prefixes from the global BGP routing table. Each prefix is checked against a set of policies, matching a single policy in accordance with the policy priority value, those with the highest value being checked first.

In version 3.9 IRP added support for policies by Country. Note that IRP maps individual prefixes to a country and does not use a transitive inference based on AS records. These prefix mappings allow IRP to more accurately work with large transcontinental AS.

Starting with version 3.9 IRP introduces a priority attribute that defines what policy to choose in cases when different policies include the same unpacked specific prefix. The policy with the highest priority will apply for such a prefix.

Note that after upgrade all existing policies are assigned (implicitly) the lowest default priority of 0.

Below you can find some typical scenarios of Routing Policies usage. For instance, there may be a specific prefix that consumes a high volume of traffic and IRP redirects it through the most expensive provider due to performance considerations. At the same time you have another two less costly available providers. In this case, you could deny the expensive provider for the specific prefix and IRP will choose the best-performing provider between the remaining two. This can be achieved by applying a Deny policy to the expensive provider or an Allow policy to the less costly providers.

The table below shows which cases a specific policy can be applied in, depending on the number of providers.

No. of providers \ Policy	Allow	Deny	Static	Static Exact
1 provider	No	Yes	Yes	Yes
2 providers or more (maximum: total number of providers - 1)	Yes	Yes	No	No
All providers (VIP probing disabled)	No	No	No	No
All providers (VIP probing enabled)	Yes	No	No	No

If a Static or Static Exact Route policy is applied to a prefix, VIP probing through each of the providers is unnecessary. Regular probing will suffice for detection of a provider failure that would trigger IRP to reroute the traffic to a different provider. Therefore IRP does not allow using VIP probing within a Static or Static Exact Route policy.

Routing policies are designed to control outgoing paths for destination networks that are outside your infrastructure. This feature should not be used to manipulate your infrastructure network behavior.

Avoid setting up of policies that point to same prefix. When improvement by aggregate is enabled, multiple prefixes can point to the same aggregate and this can cause unpredictable behaviour.

If a provider is added or removed, suspended or shutdown, the routing policies are adjusted by IRP in the following way:

A new provider is added.

Policy	Result
Allow all providers (VIP probing enabled)	The new provider is automatically included into the configured policy and probed by the VIP probing mechanism.
Allow selected providers only	The new provider is automatically ignored by the configured policy and not probed by the probing mechanism.
Deny	The new provider is automatically included into the configured policy and probed by the selected probing mechanism.
Static Route	The new provider is automatically ignored by the configured policy.
Static Exact	The new provider is automatically ignored by the configured policy.

The provider under the policy is removed.

Policy	Result
Allow all providers (VIP probing enabled)	The new provider is automatically removed from the configured policy and not probed by the VIP probing mechanism. If there is only one provider left, the policy is automatically deactivated.
Allow selected providers only	The provider is automatically removed from the configured policy and not probed by the probing mechanism. If there is only one provider left, the policy is automatically deactivated.
Deny	The provider is automatically removed from the configured policy and not probed by the probing mechanism. If there is no any other provider under this policy, it is automatically deactivated.
Static Route	The policy is automatically deactivated.
Static Exact Route	The policy is automatically deactivated.

The provider under the policy is suspended.

Policy	Result
Allow all providers (VIP probing enabled)	The provider is temporarily removed from the configured policy and not probed by the VIP probing mechanism. If there is only one provider left, the policy is temporarily deactivated.
Allow selected providers only	The provider is temporarily removed from the configured policy and not probed by the probing mechanism. If there is only one provider left, the policy is temporarily deactivated.
Deny	The provider is temporarily removed from the configured policy and not probed by the probing mechanism. If there is no any other provider under this policy, it is temporarily deactivated.
Static Route	The policy is temporarily deactivated.
Static Exact Route	The policy is temporarily deactivated.

The provider under the policy is shutdown.

Policy	Result
Allow all providers (VIP probing enabled)	The provider is temporarily removed from the configured policy and not probed by the VIP probing mechanism. If there is only one provider left, the policy is temporarily deactivated.
Allow selected providers only	The provider is temporarily removed from the configured policy and not probed by the probing mechanism. If there is only one provider left, the policy is temporarily deactivated.
Deny	The provider is temporarily removed from the configured policy and not probed by the probing mechanism. If there is no any other provider under this policy, it is temporarily deactivated.
Static Route	The policy is temporarily deactivated.
Static Exact Route	The policy is temporarily deactivated.

Policies that target an AS can be cascaded. Cascading applies the same policy to AS that are downstream to target AS, i.e. to AS that are transited by target AS.

⚠ The use case of cascading is to apply a policy to a remote AS that transits a few other AS. Still, a cascading policy can cover a huge number of down-streams. This number is parameterized and can be set to values that best fit customer's needs. Refer for example 4.4.21.

When multiple routing domains are configured a policy can be configured to prevent global improvements. Refer to Optimization for multiple Routing Domains for further details about routing domains.

Policies can be assigned a specific community and all policy based improvements will be marked with the designated value to allow their further manipulation on edge routers.
All Routing Policies are stored in /etc/noction/policies.conf file, which is automatically generated by the system.

Do not alter the /etc/noction/policies.conf file manually because any modifications will be overwritten by the system. Any changes can only be performed from the Configuration -> Routing Policies section in the system Frontend.

The rule will be ignored by the platform if the Routing Policy contains a syntax/logical error.

Please check (Routing Policy) for a detailed description of Routing Policies parameters.

1.2.10 Support for Centralized Route Reflectors

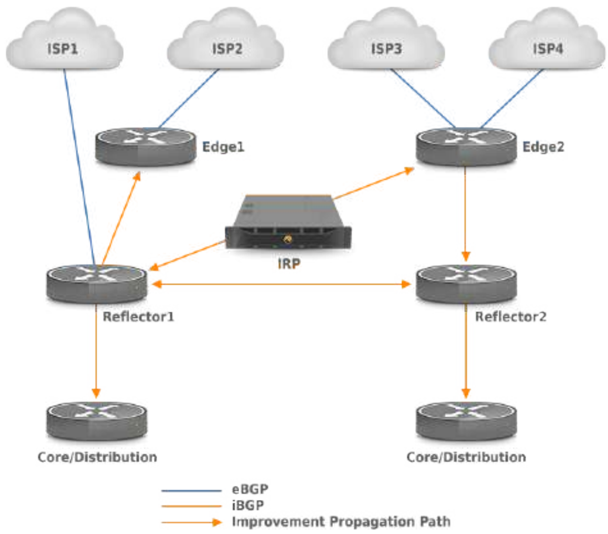

Figure 1.2.4: Support for Centralized Route Reflectors

IRP gives the possibility to advertise routes into one or more route reflectors which subsequently advertise improvements into upper and/or lower routing layers (such as edge, core or distribution).
In case the iBGP sessions can't be established between the IRP appliance and edge routers, a route reflector is used. The following restrictions apply for such a solution:

- The Next-Hop-Self option should not be used. A direct iBGP session is required between IRP and each of the divergence points (Reflector1, Edge2) in case it is enabled. This restriction is not applicable between reflectors and core/distribution layers.
- Next-Hop addresses should be accessible (exist in the routing table) where next-hop-self is not applied (Either static routes or IGP is used).
- An Internal Monitor should be configured to retrieve the eBGP session state from the device where the corresponding eBGP session is terminated. For example, ISP1 should be monitored on Reflector1, ISP2 on Edge1, and ISP3 and ISP4 on Edge2.
- Injecting routes to reflector(s) can cause temporary routing loops.

In order to announce improvements into route reflector, it should be configured as a BGP router in the “Configuration”→”BGP and routers” section and should be assigned to all the related providers in the “Configuration”→”Providers and Peers” section.

1.2.11 Support for Internet Exchanges

A transit provider can deliver traffic to any destination on the Internet. However, within an Internet Exchange, a peering partner gives access only to the set of prefixes originated or transiting its network. Therefore, when IRP evaluates the Exchange as a best path, it has to know the prefixes announced by each peer, to avoid inefficient probing of paths that cannot lead to the desired destination.

With this purpose, IRP gets the routing table from the edge router containing the list of IPs and the corresponding next-hop; this represents the next router's IP address to which a packet is sent as it traverses a network on its journey to the final destination. IRP matches the prefix with the corresponding next-hop among the configured peers, allowing it to select for probing only those peers that have access to a specific prefix. This process is also performed in the case of a transit provider that gives access only to a limited set of prefixes, rather than the entire Internet.

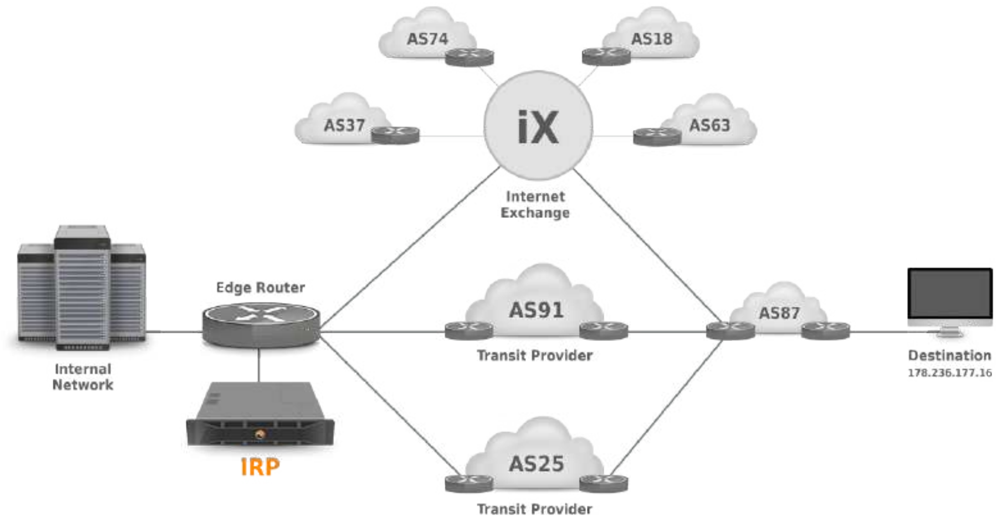

Figure 1.2.5: IRP configuration in a multi-homed network connected to transit providers as well as and Internet Exchange

In the case of multiple transit providers, there is an additional IP alias added on the IRP platform for each provider. The edge router is configured in such a way that traffic originating from each of these IPs is routed over different providers. This is done with the help of Policy Based Routing (PBR) or Flowspec policies.

With PBR, a network engineer has the ability to dictate the routing behavior based on a number of different criteria other than the destination network. These PBR rules are applied to make sure that IRP probes are following the desired paths. However, when it comes to Internet Exchanges, configuring hundreds of IP aliases on the platform would result in inefficient IP address usage and an unmanageable setup.

To avoid this, a set of PBR rules are applied making sure that the probes to be sent through a specific provider are originating from one of the configured IPs with a specific DSCP code assigned. DSCP - Differentiated Services Code Point - is a field in an IP packet that enables different levels of service to be assigned to network traffic. Since DSCP can take up to 64 different values, one configured IP can be associated with up to 64 peers. Although, due to this mechanism, the number of required IP addresses for aliases to be configured has decreased considerably, hard work would still be needed to configure the PBR on the edge router as described above.

To solve this, IRP implemented a built-in PBR config-generator which provides the configuration code to be used for a specific router model. By running this generated set of commands, network administrators can easily configure the required PBR rules on the router.

1.2.12 Optimization for multiple Routing Domains

Overview

Some networks have multiple Points of Presence interconnected both internally via inter-datacenter links and externally via multiple transit providers. The diagram below depicts an example diagram with the available routes to one destination on the Internet.

IRP uses the concept of Routing Domains to separate the locations. A Routing Domain's main characteristic is that its routing tables are mainly built on data received from its locally connected providers and the preferred routes are based on locally defined preferences.

The process of optimizing outbound network traffic in such a configuration is to mainly find better alternative routes locally (within a Routing Domain) and only reroute the traffic to other Routing Domains via inter-datacenter links when local routes are completely underperforming.

It must be noted that a multiple Routing Domain configuration works best if the Points of Presence are not too far away (ex. a network with POPs in San Francisco, Palo Alto and Danville is perfectly suitable under this scenario.

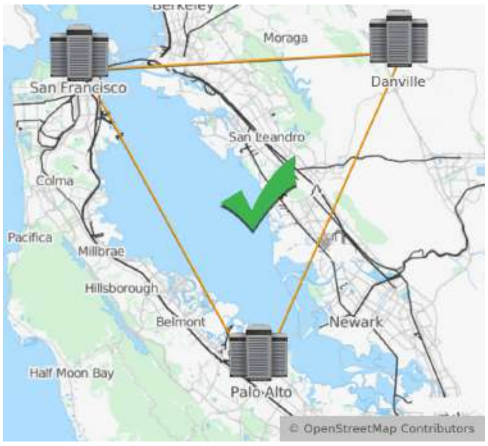

Figure 1.2.6: City wide network

POPs situated at larger distances, for example in Las Vegas and Salt Lake City are still supported by a single IRP instance running in San Francisco.

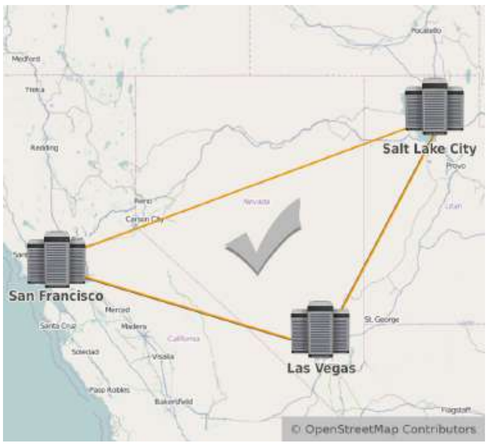

Figure 1.2.7: Regional network

Intercontinental links for POPs in Tokyo and Melbourne are way too far away from the IRP instance in San Francisco and in such a case multiple IRP instances are required.

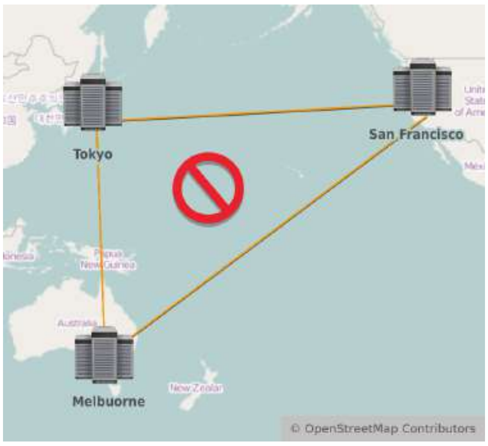

Figure 1.2.8: Intercontinental network

Multiple Routing Domains implementation attributes To further detail the multiple routing domain attributes the following diagram will be used:

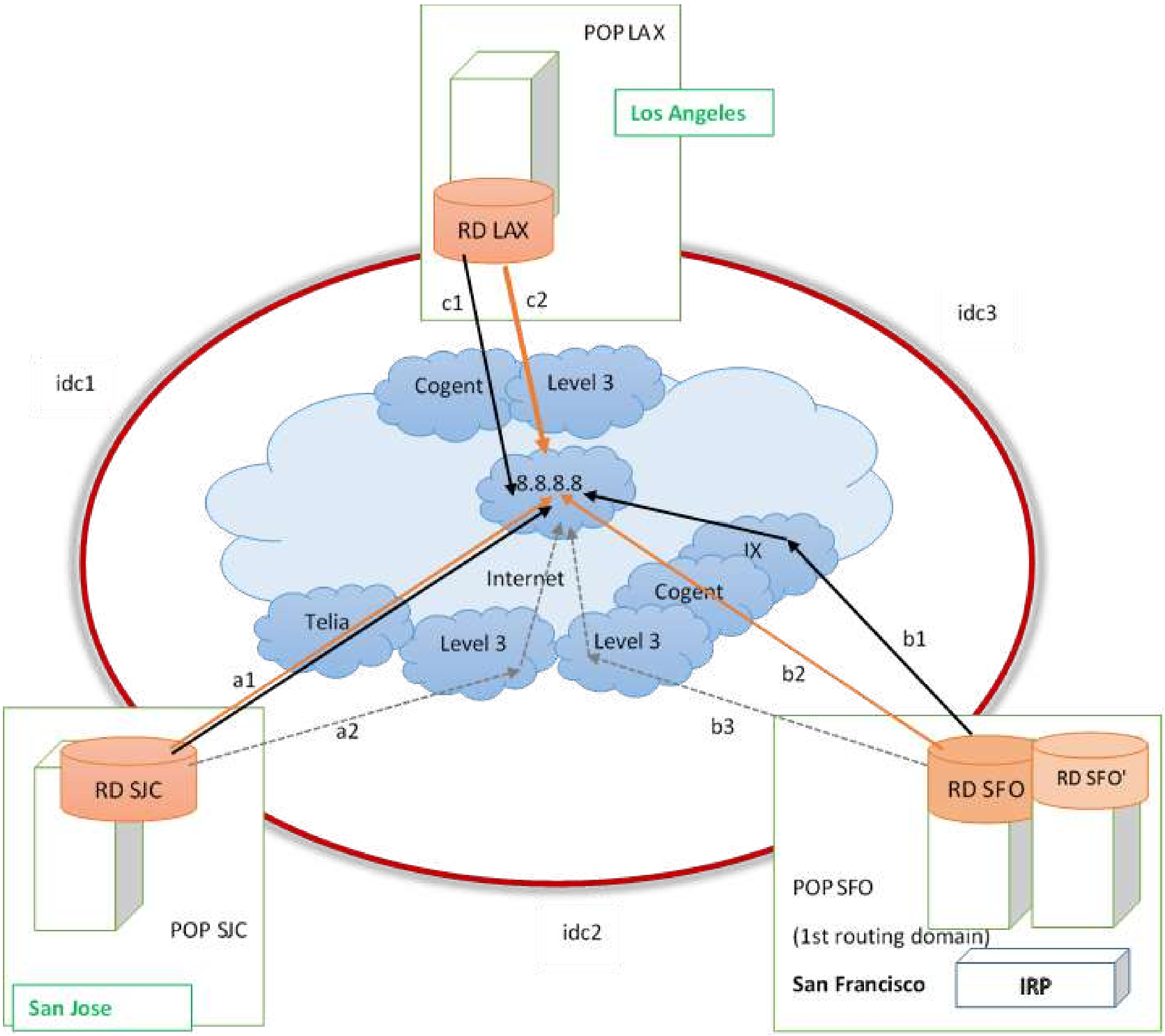

Figure 1.2.9: Multiple routing domains

A multiple Routing Domain configuration has a series of attributes:

- Multiple locations belonging to the same network (AS) represented in the diagram by POP SJC, POP SFO and POP LAX (of course, more than 3 routing domains are supported).
- The locations are distinguished by the different Routing Domains within which they operate (depicted by RD SJC, RD SFO, and RD LAX)
- The Routing Domains are managed by edge routers belonging to different locations
- Nearby locations that process routing data differently should be split into different Routing Domains, even if they have the same upstream providers. In the diagram above RD SFO and RD SFO' are depicted as part of a single Routing Domain. A decision to split or keep in the same routing domain should be made based on exact knowledge on how routing data is processed.
- Inter-datacenter loop interconnects the different locations (depicted by idc1, idc2 and idc3 segments)
- Data flows between locations take only the short path (in the example POP SJC can be reached from POP SFO via idc2 path (short) or idc3 + idc1 path (long))
- Each Routing Domain has different providers and different preferred routes to reach a specific destination (a1, b1, c1)
- A single IRP instance collects statistics about traffic (Irpflowd only), probes available destinations and makes improvements towards specific prefixes/networks on the Internet.
- IRP assumes RTT of zero and unlimited capacity to route traffic within a Routing Domain
- IRP assumes that Sites are not physically too far away. It is ok to have different sites in the same city or region as at this scale inter-datacenter links have predictable characteristics. When taking intercontinental links into consideration this is quite probably not the case.
- Distances between sites (idc1, idc2, idc3 delays) are measured in advance and specified in IRP's configuration.

Inter-datacenter link characteristics Support for Multiple Routing Domains relies on existence of inter-datacenter links. These links should be independent of upstream providers.
Example of inter-datacenter links that multiple routing domains is designed for are:

- private connections,
- L2 links with guaranteed service,
- MPLS links

⚠ VPNs via public Internet could be used with Multi Routing Domain feature but is a suboptimal choice. Under such conditions IRP MUST be prevented from making Global Improvements. This way IRP will do only local optimizations in each Routing Domain and will operate similarly to multiple IRP instances (while probing excessively because it probes destinations via remote locations too).

Constraints At the moment IRP multiple Routing Domains implementation does not cover the following:

- IRP does not take measurements of inter-datacenter link delays (idc1, idc2 and idc3). This values are configurable.
- IRP does not monitor if inter-datacenter links are operating normally. In case such a link is broken it is expected IRP to loose BGP connectivity with routing domain routers and this will cause IRP improvements to be withdrawn till the link is restored.

- IRP does not try to detect if the traffic is following long or short paths on the inter-datacenter links. In the image above traffic from RD SJC can follow path idc1 (short) or idc2+idc3 (long). IRP always assumes the short path is being followed internally.
- IRP does not take measurements of inter-datacenter link capacity and current bandwidth usage. At this stage IRP assumes there is enough inter-datacenter link capacity to also carry the (few) global improvements. Also, IRP tries to minimize usage of inter-datacenter links.

Routing domains

Routing domain is a generic term used to distinguish a logical location that works with different routing tables. The differences are caused by the fact that a router composes its routing table according to routes received from different providers. It is possible to have multiple routing domains in the same datacenter if routing data is received by different routers (even from same or different sources) and data flows are distributed via different routers by different policies. In the image above RD SFO and RD SFO' can represent a single routing domain or multiple routing domains depending on what routing policies are applied.

Different routing domains are assigned identifiers in the range 1-100. Routing Domain identifier is assigned individually to each provider via parameter peer.X.rd. It must be noted the Routing domain that hosts the IRP instance is considered as the first routing domain (RD=1).

Parameter global.rd_rtt gives the distances between routing domains. The format of the parameter is

```
rda:rdb:rtt
```

for example if RD SJC has Routing Domain id = 42, RD SFO - 1 (since it hosts IRP), RD LAX - 3 then the idc1, idc2 and idc3 rtt is defined as the collection:

```
global.rd_rtt = 3:42:20 42:1:17 1:3:35
```

This parameter will be validated for correctness and besides the format above it requires that RD SJC and RD SFO values are different and already configured (RD1 is always present).

Round trip time between one routing domain and another is calculated by executing PING towards edge routers and taking average integer value:

```
$ ping X -c 10 -q
PING X (X) 56(84) bytes of data.
--- X ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9085ms
rtt min/avg/max/mdev = 40.881/41.130/41.308/0.172 ms
```

Flow agents

A very natural constraint for Multiple Routing Domain networks is that IRP can rely only on Flow statistics - NetFlow or sFlow.

SPAN cannot be used because it does not carry attributes to distinguish traffic between different providers

Flow collector needs to know the exact details of such a configuration in order to correctly determine the overall provider volume and active flows. For this each provider in an MRD setup must be assigned Flow agents to enable IRP to match Flow statistics accordingly. Refer Flow agents for further details.

Global and local improvements

Local improvements Local improvements represent better alternative routes identified within a routing domain. If in the example image above current routes are represented by black lines then local improvements are depicted by orange lines b2 and c2. Keep in mind that a1 just reconfirmed an existing current route and no improvements are made in such a case.

Local improvements are announced in their routing domains and this has the characteristic that local traffic exits customer's network via local providers. This also means that inter-datacenter interconnects are free from such traffic.

IRP prefers local routes and Improvements to Global improvements.

Parameter bgpd.rd_local_mark specifies a community marker that distinguishes local improvements from Global Improvements. A BGP speaker should not advertise these improvements outside its Routing Domain. It must be noted that a single marker is used for all the routing domains and each of them shall be configured to advertise local improvements within the domain and filter it out for inter-domain exchanges.

Local improvements should be stopped from propagating across routing domains. A route map is used to address this. Below are listed sample route maps for Cisco IOS and JUNOS 9.

Cisco IOS

Refer your router capabilities in order to produce the correct route map. The route map MUST be integrated into existing route maps. It is not sufficient to simply append them.

```
neighbor <neighbor-from-another-RD> send-community (should be configured
   for all iBGP sessions)

ip community-list standard CL-IRP permit 65535:1
route-map RM-IRP-RD deny 10
 match community CL-IRP
route-map RM-IRP-RD permit 20

router bgp AS
 neighbor <neighbor-from-another-RD> route-map RM-IRP-RD out
```

Refer Route-Maps for IP Routing Protocol Redistribution Configuration

JUNOS 9

Refer your router capabilities in order to produce the correct route map. The route map MUST be integrated into existing route maps. It is not sufficient to simply append them.

```
policy-options{
      policy-statement IRP-CL {
            term 0 {
            from {
                  protocol bgp;
                  community IRP-RD;
            }
            then reject;
         }
         term 1 {
            then accept;
         }
      }
      community IRP-RD members 65535:1;
}
```

```
protocols {
      bgp {
      group ebgp {
            type external;
            neighbor 10.0.0.1 {
            export IRP-CL;
            }
      }
   }
}
```

Refer Policy Framework Configuration Guide; Release 9.3

Global improvements Global improvements are made when IRP identifies an alternative route that

even after factoring in the latencies incurred by inter-datacenter interconnects are better than all existing alternatives. Such an example can be represented by alternative route c2 in the image above. A global improvement is made when one routing domain alternative is better than the best local alternatives in all other routing domains even considering the latencies incurred by inter-datacenter interconnects. In the image above c2 will become a global improvement if his loss characteristic is best to all alternatives and its latency:

- (c2+idc1 - margin) is better than best local alternative a1 in RD SJC
- (c2+idc3 - margin) is better than best local alternative b2 in RD SFO

where:

- a1, b2 and c2 represent roundtrip times determined by IRP during probing of a destination.
- idc values are configurable and are set as one entry of global.rd_rtt parameter.
- margin is given by core.global.worst_ms.

> Global improvements can degrade performance in some routing domains. If performance for some routing domains degrades, IRP announcements for the global improvement also carry designated BGP community attributes set by rd.X.community_worsening.

Global improvements move traffic via inter-datacenter interconnects and as such are less desirable to local routes. Global improvements make sense when defined as above and even more sense when packet loss is taken in consideration and routing via a different datacenter reduces packet loss significantly.

1.2.13 Improvements weight

IRP assigns each improvement a weight. The weight takes into consideration many network and environment aspects of the change such as policy or VIP destinations, loss and latency differences, cost or commit control type of the improvement. Based on all of the above the improvement gathers more or less weight as appropriate.

Later on, instead of replacing oldest improvements that might still bring significant benefits with new improvements just because they are fresh, IRP relies on the weights to decide whether the benefit of the new improvement is sufficient to replace an existing one. More so, besides preserving the most relevant improvements this feature reduces route flapping by blocking announcement of new improvements and withdrawal of existing ones if the changes are not offering a good enough return.

1.2.14 Notifications and events

IRP produces a huge number of various events and some of them are critical for customer's awareness. Notifications allow customers to subscribe to any of the available events using the following channels:

- SMS
- Email
- Slack (via Webhook)
- SNMP Traps

IRP service Irppushd provides this feature. In order for Notifications to be delivered correctly the corresponding channel configuration shall be provided. By default only email notifications can be delivered since IRP uses the embedded system email service to send them.

More so, users should subscribe for specific events.

Only events for valid subscriptions using correctly configured channels will be delivered.

Refer section IRP instance Notifications for details about configuring, subscribing and contents of notifications.

Refer section Notification and events for details about individual configuration parameter.

Events

The list of events monitored by IRP that can generate notifications is provided below.

When one of the IRP components detects a transition form normal to abnormal traffic behavior or back it fires these events:

- Abnormal correction: irpflowd
- Abnormal correction: irpspand
- Inbound traffic low: SPAN
- Inbound traffic low: Flow
- Inbound traffic normal: Flow
- Inbound traffic normal: SPAN
- Outbound traffic low: SPAN
- Outbound traffic low: Flow
- Outbound traffic normal: Flow
- Outbound traffic normal: SPAN

When Commit Control limits are exceeded per provider or overall one of the following events fires. Refer section 4.12 for configuring the actual limits of the events.

- Commit Control overload by X Mbps
- Commit Control overload by X%
- Commit Control provider X overloaded by Y Mbps
- Commit Control provider X overloaded by Y%

When an IRP component (re)loads the configuration it validates it and depending on results fires one of the following events:

- Configuration Invalid: Bgpd
- Configuration Invalid: Core
- Configuration Invalid: Explorer
- Configuration Invalid: Irpapid
- Configuration Invalid: Irpflowd
- Configuration Invalid: Irpspand
- Configuration Ok: Bgpd
- Configuration Ok: Core
- Configuration Ok: Explorer
- Configuration Ok: Irpapid
- Configuration Ok: Irpflowd
- Configuration Ok: Irpspand

Outage detection algorithm fires one of the following events when it confirms congestion or outage problems and reroutes traffic around it:

- Congestion or Outage
- Outage: Confirmed and rerouted

Explorer periodically checks the PBRs and its expected probing performance and triggers the following events:

- Failed PBR (IPv6) check for provider
- Failed PBR (IPv4) check for provider
- Successful PBR (IPv4) check for provider
- Successful PBR (IPv6) check for provider
- Explorer performance low
- High number of VIP prefixes degrades IRP performance

IRP BGP Internal and External monitors fire the following events:

- ExternalMonitor (IPv4) Failed status for a provider. All improvements towards the provider will be withdrawn.
- ExternalMonitor (IPv4) OK status for a provider. All improvements towards the provider will be announced.
- ExternalMonitor (IPv6) Failed status for a provider. All improvements towards the provider will be withdrawn.
- ExternalMonitor (IPv6) OK status for a provider. All improvements towards the provider will be announced.
- InternalMonitor (IPv4) Failed status for a provider. All improvements towards the provider will be withdrawn.

- InternalMonitor (IPv4) OK status for a provider. All improvements towards the provider will be announced.
- InternalMonitor (IPv6) Failed status for a provider. All improvements towards the provider will be withdrawn.
- InternalMonitor (IPv6) OK status for a provider. All improvements towards the provider will be announced.

When statistics collection over SNMP is up or down IRP fires the following events:

- Provider SNMP stats down: X
- Provider SNMP stats up: X

Bgpd raises these events when BGP sessions are established/disconnected:

- IRP BGP session disconnected
- IRP BGP session established

When IRP identifies conditions to re-route traffic (make an improvement) and additionally it considers the differences to be excessive it raises these events:

- Excessive packet latency for prefix
- Excessive packet loss for prefix
- Improvements spike
- Low rate of announced IPv4 improvements
- Low rate of announced IPv6 improvements
- New improvement

Once an IRP component is started, stopped or restarted it raises the following events:

- Service started: Bgpd
- Service started: Core
- Service started: Explorer
- Service started: Irpapid
- Service started: Irpflowd
- Service started: Irpspand
- Service stopped: Bgpd
- Service stopped: Core
- Service stopped: Explorer
- Service stopped: Irpapid
- Service stopped: Irpflowd
- Service stopped: Irpspand

SNMP Traps

SNMP traps is a widely used mechanism to alert about and monitor a system's activity.

IRP SNMP traps not only notify about some IRP platform event but also include the list of varbinds which contain detailed information related to the thrown trap. The complete list of traps and varbinds with their descriptions can be found at `/usr/share/doc/irp/NOCTION-IRP.mib`

1.2.15 IRP API

IRP exposes a web API that uses HTTP verbs and a RESTful endpoint structure. Request and response payloads are formatted as JSON.

The API is running on the IRP instance and is reachable by default over SSL at port 10443. If called directly from the IRP instance server the API can be accessed at https://localhost:10443 Use https://hostname:10443/ in order to access the API from elsewhere on the network.

An IRP user id is required to access most of the API services. Use GMI to manage user API access tokens. IRP API uses an authenticating mechanism based on authentication tokens. The token is passed as a query parameter for all API requests that require authentication.

The API Reference is available from GMI menu.

IRP's API is powered by Irpapid service that can be started, stopped, configured like any other IRP service. Refer to the 4.3 section for Irpapid configuration parameter details.

1.2.16 IRP Failover

Overview

IRP offers failover capabilities that ensure Improvements are preserved in case of planned or unplanned downtime of IRP server.

IRP's failover feature uses a master-slave configuration. A second instance of IRP needs to be deployed in order to enable failover features. For details about failover configuration and troubleshooting refer Failover Configuration.

A failover license is required for the second node. Check with Noction's sales team for details.

IRP's failover solution relies on:

- slave node running same version of IRP as the master node,
- MySQL Multi-Master replication of 'irp' database,
- announcement of the replicated improvements with different LocalPref and/or communities by both nodes,
- monitoring by slave node of BGP announcements originating from master node based on higher precedence of master's announced prefixes,
- activating/deactivating of slave IRP components in case of failure or resumed work by master,
- syncing master configuration to slave node.

For exact details about IRP failover solution refer to configuration guides (2.13, 3.2.1.4), template files, and (if available) working IRP configurations. For example, some 'irp' database tables are not replicated, 'mysql' system database is replicated too, some IRP components are stopped.

IRP versions 3.5 and earlier do no offer failover capabilities for Inbound improvements. It is advised that in these versions only one of the IRP instances is configured to perform inbound optimization in order to avoid contradictory decisions. In case of a failure of this instance inbound improvements are withdrawn.

An overview of the solution is presented in the following figure:

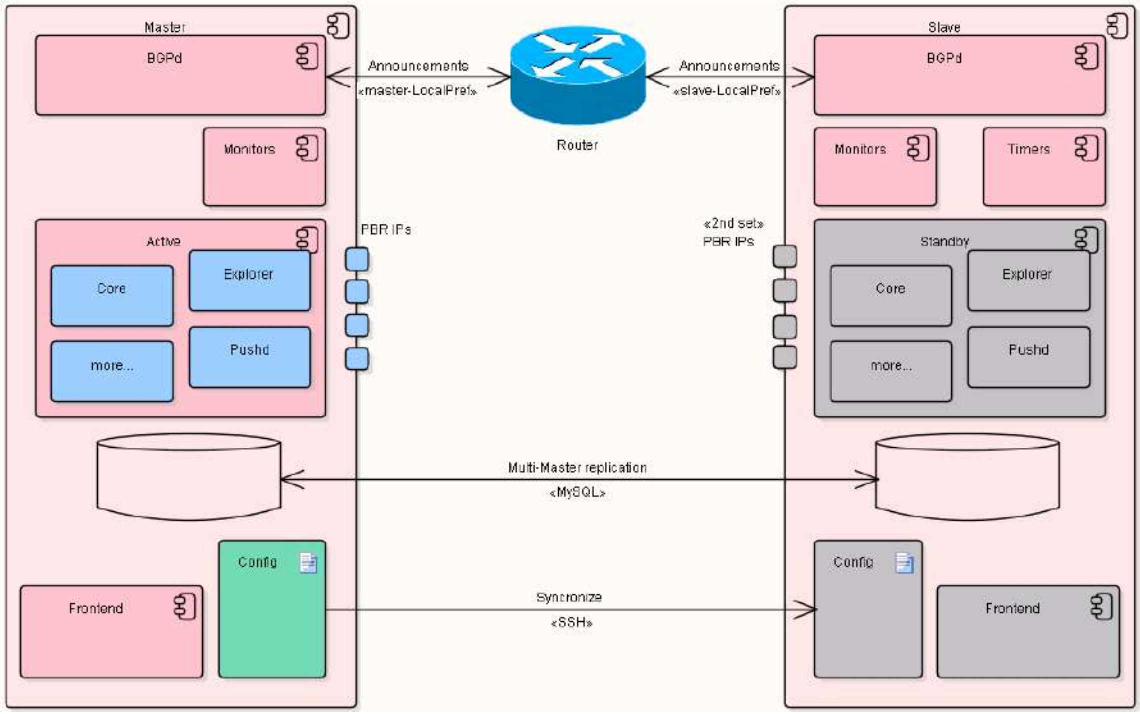

Figure 1.2.10: Failover high level overview

The diagram highlights the:

- two IRP nodes - Master and Slave,
- grayed-out components are in stand-by mode - services are stopped or operating in limited ways. For example, the Frontend detects that it runs on the slave node and prohibits any changes to configuration while still offering access to reports, graphs or dashboards.
- configuration changes are pushed by master to slave during synchronization. SSH is used to connect to the slave.
- MySQL Multi-Master replication is setup for 'irp' database between master and slave nodes. Existing MySQL Multi-Master replication functionality is used.
- master IRP node is fully functional and collects statistics, queues for probing, probes and eventually makes Improvements. All the intermediate and final results are stored in MySQL and due to replication will make it into slave's database as well.
- Bgpd works on both master and slave IRP nodes. They make the same announcements with different LocalPref/communities.
- Bgpd on slave node monitors the number of master announcements from the router (master announcements have higher priority than slave's)
- Timers are used to prevent flapping of failover-failback.

Requirements

The following additional preconditions must be met in order to setup failover:

1. second server to install the slave,
2. MySQL Multi-Master replication for the irp database.

MySQL replication is not configured by default. Configuration of MySQL Multi-Master replication is a mandatory requirement for a failover IRP configuration. Failover setup, and specifically MySQL Multi-Master replication should follow a provided failover script. Only a subset of tables in irp database are replicated. Replication requires extra storage space, depending on the overall traffic and platform activity, for replication logs on both failover nodes.

1. a second set of BGP sessions will be established,
2. a second set of PBR IP addresses are required to assign to the slave node in order to perform probing,
3. a second set of improvements will be announced to the router,
4. a failover license for the slave node,
5. Key-based SSH authentication from master to slave is required. It is used to synchronize IRP configuration from master to slave,
6. MySQL Multi-Master replication of 'irp' database,
7. IRP setup in Intrusive mode on master node.

In case IRP failover is setup in a multiple Routing Domain configuration and IRP instances are hosted by different RDs this must be specified in IRP configuration too. Refer Optimization for multiple Routing Domains, global.master_rd, global.slave_rd.

Failover

IRP failover relies on the slave node running the same version of IRP to determine if there are issues with the master node and take over if such an incident occurs.

Slave's Bgpd service verifies that announcements are present on a router from master. If announcements from master are withdrawn for some reason the slave node will take over.

In order for this mechanism to work IRP needs to operate in Intrusive mode and master's node announcements must have higher priority then the slave's.

During normal operation the slave is kept up to date by master so that it is ready to take over in case of an incident. The following operations are performed:

- master synchronizes its configuration to slave. This uses a SSH channel to sync configuration files from master to slave and process necessary services restart.
- MySQL Multi-Master replication is configured on relevant irp database tables so that the data is available immediately in case of emergency,
- components of IRP such as Core, Explorer, Irppushd are stopped or standing by on slave to prevent split-brain or duplicate probing and notifications,
- slave node runs Bgpd and makes exactly the same announcements with a lower BGP LocalPref and/or other communities thus replicating Improvements too.

It is imperative that master's LocalPref value is greater than slave's value. This ensures that master's announcements are preferred and enables slave to also observe them as part of monitoring.

In case of master failure its BGP session(s) goes down and its announcements are withdrawn.

Slave node only considers that master is down and takes over only if master's Improvements are withdrawn from all edge routers in case of networks with multiple edge routers.

The same announcements are already in router's local RIB from slave and the router chooses them as best.

This is true only if LocalPref and/or communities assigned to slave node are preferred. If other most preferable announcements are sent by other network elements , no longer announcements from slave node will be best. This defeats the purpose of using IRP failover.

At the same time, Failover logic runs a set of timers after master routes are withdrawn (refer global.failover_timer_fail). When the timers expire IRP activates its standby components and resumes optimization.

Failback

IRP includes failback feature too. Failback happens when master comes back online. Once Bgpd on the slave detects announcements from master it starts its failback timer (refer global.failover_timer_failback). Slave node will continue running all IRP components for the duration of the failback period. Once the failback timer expires redundant slave components are switched to standby mode and the entire setup becomes normal again. This timer is intended to prevent cases when master is unstable after being restored and there is a significant risk it will fail again.

During failback it is recommended that both IRP nodes are monitored by network administrators to confirm the system is stable.

Recovery of failed node

IRP failover configuration is capable to automatically restore its entire failover environment if downtime of failed node is less than 24 hours.

Recovery speed is constrained by restoring replication of MySQL databases. On 1Gbps non-congested links replication for a full day of downtime takes approximately 30-45 minutes with 200-250Mbps network bandwidth utilization between the two IRP nodes. During this time the operational node continues running IRP services too.

If downtime was longer than 24 hours MySQL Multi-Master replication is no longer able to synchronize the databases on the two IRP nodes and manual MySQL replication recovery is required.

Upgrades

Failover configurations of IRP require careful upgrade procedures especially for major versions.

It is imperative that master and slave nodes are not upgraded at the same time. Update one node first, give the system some time to stabilize and only after that update the second node.

1.2.17 Inbound bandwidth optimization

1.2.17.1 Inbound Commit Control

Starting with version 3.4 IRP introduced optimization of Inbound traffic. Inbound bandwidth control reshapes the traffic from different providers targeting your sub-prefixes .

IRP uses well known and proven BGP mechanisms to instruct your routers to adjust their advertisements of your network segments to upstream providers and subsequently to the World. The adjusted advertisements take advantage of existing BGP policies implemented by edge routers worldwide in order to increase or decrease the preference of your internal network segments as advertised by one or another AS to the world. This allows more traffic to lean towards some of your upstream providers and less towards others. In case of an incident that your multihomed configuration is designed to be resilient against, the entire world still knows of the alternative routes towards your network and will be able to adjust accordingly.

Noction's IRP Inbound feature:

- advises your edge routers to advertise different prefixes of your network with different counts of BGP prepends to each of your upstream providers;
- monitors inbound and outbound traffic on your network interfaces using standard protocols (SNMP, NetFlow, sFlow) to determine if there is need for action;
- continuously assesses network health and performance to ensure that when overloads are possible it knows good and reliable alternative routes to use;
- uses a proprietary inferring mechanism that takes into account the inertial nature of the Internet to react to inbound preference changes and thus dampens the urge to "act now" and destabilize the network;
- provides you with configuration, monitoring and visualization tools that allow you to cross-check and validate its actions.

The entire Inbound optimization solution covers:

1. Segmenting your network into prefixes that are controlled by IRP
2. Coordinating communication over BGP between IRP and routers
3. Setting network conditions for making Inbound Improvements
4. Reviewing Inbound optimization results

Starting with version 3.7 IRP has the capability to also monitor and improve transit routes. Refer Optimization of transiting traffic for details.

The use cases below highlight typical scenarios when IRP's inbound bandwidth control capabilities might come handy:

Lets start with the case of unbalanced inbound and outbound peering You have a peering agreement with one of your neighbors to exchange traffic. Unfortunately the rest of the neighboring network configuration significantly unbalances the volumes of inbound and outbound traffic.

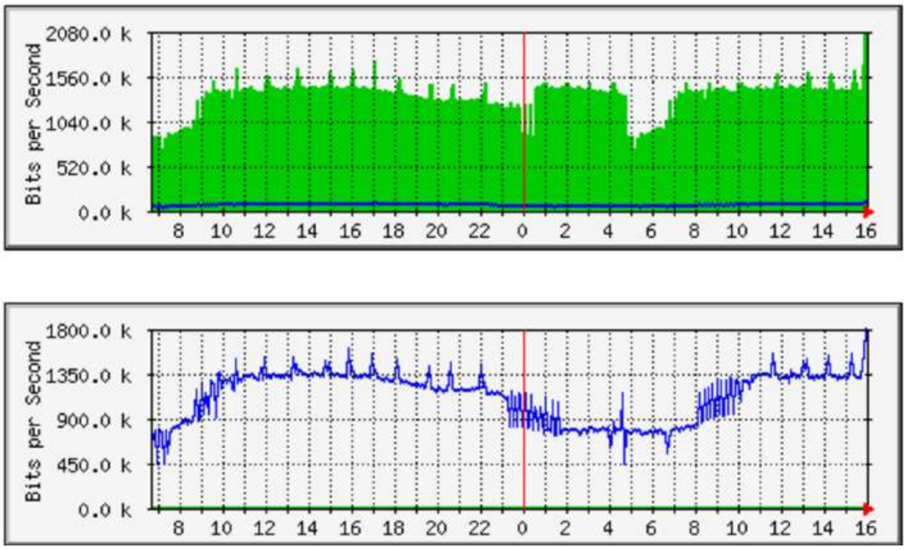

Figure 1.2.11: Unbalanced inbound/outbound peering

You rely on manipulating prefix announcements towards neighbors in order to shape the traffic. Unfortunately this is a reactive solution and consumes a lot of time while at the same time pushing the balance either one way or another. Often this pushes the network into the second typical scenario.

Fluctuating traffic shape A multihomed configuration overwhelms some links while other links remain barely used. Your network administrators frequently get alerts during peak network use and they manually add or remove prepends or altogether remove some inbound prefixes from being announced to affected neighboring links. These changes are sometime forgotten or other times just push the bulk of traffic towards other links and the problem re-appears.

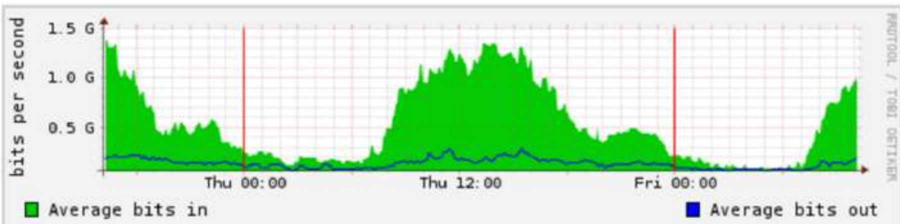

Figure 1.2.12: Fluctuating inbound traffic

For both above scenarios Inbound commit control in IRP can automate most of the inbound traffic shaping operations. It uses a typical solution for traffic shaping and specifically manipulates prepend counts announced to different providers and this eventually propagates across the Internet thus diverting traffic as desired. IRP automates your routine day to day traffic shaping actions and probably makes significantly more adjustments than you can afford to. At the same time it offers you reliable reviewing and fine-tuning options that will allow you to adjust IRP's work in a changing and evolving network.

Refer Inbound Commit Control, Current Inbound Improvements, Inbound Traffic Distribution, Inbound rule for details.

1.2.17.2 Inbound Performance Optimization

1.2.18 Inbound performance optimization

Starting with version 4.0 IRP is able to perform loss and latency optimization of inbound traffic.

This feature allows improving the inbound traffic performance by deflecting it from the worst-performing provider. The improvement actions do not guarantee the best path for the inbound traffic but minimize the usage of the worst path. For the traffic deflection consistency, IRP uses provider traffic engineering capabilities.

Some network providers give customers the ability to influence their traffic based on the announced BGP community, the so-called traffic engineering capability. Customers of such providers can check if the upstreams support this functionality by reviewing the upstreams BGP routing policies, formerly published in Whois/RDAP for provider's Autonomous System, or by contacting providers directly. In other words, traffic engineering capability, therefore, represents a list of BGP communities that a provider configures on their side for the customer's use. When prefixes tagged with a specific BGP community from such lists get announced by clients, the upstream provider applies the corresponding action to those prefixes.

The requirements for the Inbound performance feature are:

- Provider traffic engineering capability using BGP community
- Ability to re-announce providers' BGP table to IRP. Configuring the BGP monitoring Protocol (BMP BMP monitoring station) is recommended
- Flow export (The feature is not supported in the SPAN-only setups)

Calculation of an inbound performance improvement takes more time, when compared to the regular outbound improvement. The reason is that the inbound traffic is affected much slower than the outbound traffic and the calculation method is different.

For the Inbound performance feature to work, IRP must know what traffic should be monitored and improved. For that, the Inbound Performance rule(s) must be configured. More about Inbound performance rules: Inbound Performance Rules

IRP performs the analysis of network performance indicators and applies improvements per individual preconfigured rules in the following order:

- Probes prefixes that generate inbound traffic to build a statistical model of the route performance per each inbound rule
- Reprobes prefixes according to the configured inbound reprobing interval (irpinperfd.probing.interval)
- Compares the reprobed results with the statistical model
- In case the poor inbound traffic metrics are obtained – IRP makes an improvement by announcing the customer prefixes with the corresponding TE community
- Once IRP applies an inbound improvement, its relevance is checked on subsequent probing sessions

There are two modes in which the inbound performance improvements can be enabled:

• Automated - IRP enables/disables all the inbound performance improvements automatically once an improvement is calculated

• Moderated - stands for manual confirmation of the inbound performance improvements. Once IRP detects an improvement, a suggestion to enable it gets displayed in the GMI interface. IRP does not perform any improvements without user confirmation. If an improvement is not enabled in time before a subsequent cycle of checks, IRP reviews the proposal and adjusts it as needed, keeping the recommendation, making a different one, or removing the improvement proposal

The inbound performance improvements can be enabled manually as per a particular rule. This is done at the discretion of the user. IRP does not make any calculations/proposals in such cases

To view how to configure the Inbound Performance feature, go to: Inbound performance optimization configuration

1.2.19 Flowspec policies

Flowspec policies can be used only in conjunction with Flowspec capable routers.

Starting with version 3.5 IRP has support of Flowspec policies. This means that Flowspec capability is recognized and can be used accordingly for BGP sessions established by IRP. In short Flowspec defines matching rules that routers implement and enforce in order to ensure that only desirable traffic reaches a network. Flowspec by relying on BGP to propagate the policies uses a well understood and reliable protocol.

As specified by BGP, when a session disconnects all the announcements received from that session are discarded. The same is generally true for Flowspec policies. Still, since some vendors recognize Flowspec policies but implement them using capabilities other than BGP a confirmation is needed whether on session disconnect the specific router model indeed removes all the underlying constructs and reverts to a known state.

IRP not being involved in direct packet forwarding expects that Flowspec policies are implemented at least by your edge routers. If upstream providers also offer Flowspec support these policies can be communicated upstream where their implementation is even more effective.

Eventually Flowspec policies help ensure that traffic entering or exiting your network conforms to your plans and expectations. The main use cases that can be accomplished with Flowspec policies in IRP allows:

- controlling bandwidth usage of your low priority traffic towards external services, for example throttling bandwidth usage originating on your backup systems towards off-premises services.
- anticipating inbound traffic towards your services and shaping bandwidth use in advance, for example anticipating low numbers of legitimate customers from Russia, China or India on your e-commerce services and setting high but controllable rate limits on packets originating in those networks.
- reacting on a packet flooding incident by dropping specific packets, for example dropping all packets targeting port 53.
- redirecting some traffic for scrutiny or cleansing, for example forwarding port 80 packets through an intelligent device capable of detecting RUDY, slow read or other low-bandwidth/amplification attacks.

IRP Flowspec policies rely on a minimal set matching rules and actions that offer most of the capabilities while keeping the learning curve low and integration simple:

- Source or destination IP address specified as either CIDR format prefix or direct IP address
- Traffic protocols, for example TCP, UDP or ICMP
- Source or destination TCP/UDP ports
- Throttle, drop and redirect actions.

It is important to note that IRP does not cross-validate Flowspec policies with improvements. While it is possible that for example a Flowspec redirect action pushes some traffic a different way to what an improvement advises, usually improvements cover many prefixes and while there will be a contradiction for one prefix there will be many other prefixes that IRP improves to compensate for these unitary abnormalities. It is recommended that Flowspec policies take precedence over improvements in order to benefit from this compensating nature of improvements.

Consider that depending on whether source or destination prefix belongs to your network the policy applies to either inbound or outbound traffic while the choice of ports allows targeting different traffic types.

Compare Flowspec policies to the already well known Routing Policies . For further details regarding Flowspec configuration refer Flowspec Policies.

1.2.20 Throttling excessive bandwidth use with Flowspec

IRP can be configured to automatically add throttling Flowspec policies for prefixes that started using abnormal volumes of traffic. This feature can be used only if your network has Flowspec capabilities. Refer Flowspec policies for details about Flowspec.

In case thresholds for excessive bandwidth use are set to very aggressive levels IRP can create large numbers of Flowspec policies.

This feature can be described as:

- configure excess threshold and throttling multipliers
- periodically determine current and average prefix bandwidth usage for this hour of the day
- verify if current usage exceeds the average by a larger factor then the threshold multiplier
- rate-limit excessive bandwidth usage prefixes at their average use times throttling multiplier.

Past throttling rules are revised if/when prefix abnormal usage pattern ends.

For example, when a prefix usually consumes 1-2Mbps of traffic and its current bandwidth spikes tenfold and the spike is sustained for a significant period of time a throttling rule limiting usage by this prefix at 5-6Mbps will still offer ample bandwidth for normal service use and will also protect other services from network capacity starvation.

Refer Core configuration for details.

1.2.21 Maintenance windows

Maintenance works are on everybody's agenda in current fast paced and continuously evolving networks. During maintenance network engineers are very busy and will welcome any help their systems can offer in carrying out those works with the least amount of headaches. IRP is clearly not in the top of network engineer's priorities and asking to suspend or shutdown a provider immediately before a maintenance window starts and restart the provider back once the maintenance works end is not very helpful if not even annoying.

Instead IRP offers the facility to plan maintenance windows in advance. Knowing when a maintenance window starts and ends, IRP excludes that specific provider link from either performance optimization or bandwidth control. More so, IRP has the capability to reshape the traffic flowing in and out of a network to anticipate any downtime on a link.

Setting a maintenance window by router sets each of the providers on the router with a maintenance window of their own.

Properly configured maintenance windows allows IRP time to move most of the outbound traffic and deflect most of inbound traffic away from the provider link that is scheduled for maintenance. Having only a small fraction of traffic or none at all on the maintenance link before the downtime starts avoids any (shall we say, catastrophic) spikes, possible overloads and consequently unpredictable behavior of the remaining live network equipment.

Specifically the following applies:

- a maintenance window is configured in advance and can be removed/revised at any time,
- a maintenance window sets details for single provider. If needed multiple maintenance windows can be setup and even overlapping maintenance windows are OK.
- IRP highlights maintenance windows in IRP's Frontend sidebar so that it is easy to spot current maintenance window status.
- optionally IRP can preserve existing improvements so that once the maintenance window ends improvements are reimplemented. It is advised that this feature is used only when the maintenance window is very short (a few minutes).
- an unloading period can be setup. During unloading IRP actively re-routes outbound prefixes through other available providers. While IRP is able to make most of the unloading improvements fast consideration shall be given to the announcement rate limitations setup in Bgpd in order for all the improvements to reach network routers in time for maintenance window starting time.
- a prepend time can be setup. This is only applicable if Inbound optimization is operational. If this time is setup then IRP will prepend configured inbound prefixes with the maximum allowed number of prepends through the provider link under maintenance in order to deflect inbound traffic towards other providers. Refer to Inbound bandwidth optimization for more details about Inbound optimization.

Refer Maintenance windows for details how to configure, review, create, edit, delete maintenance windows.

1.2.22 Optimization of transiting traffic

Starting with IRP 3.7 IRP introduces optimization of transiting traffic capabilities. Optimization of transiting traffic is an enhancement of Inbound bandwidth optimization .

Optimization of transiting traffic relies on the same method of influencing Internet-wide routing - manipulating best path selection by increasing AS Path length for a prefix carrying traffic on an undesirable interface. The most secure and effective way of AS Path manipulation is to apply these changes on a router facing a provider. Edge routers prepend routes according to designated communities.

Besides being an enhancement of Inbound optimization, transit prefixes are governed by a different set of constraints:

- The number of potential routes is very large and only some will/should be targeted for optimization. For this IRP uses filters by ASN/prefix allowing network administrators to configure what segments of the Internet to focus IRP's attention to.
- The improvements are visible on the Internet and excessive route changes can be flagged by external monitoring services as flapping or route instability. IRP protects against this by rate limiting number of route changes through a specific provider.
- Once a route has been improved all its traffic might be diverted away from this network and no new statistics will be available to make further inferences. In such cases IRP reverts old transit improvements during network's off-peak hour by either decreasing the number of advertised prepends or withdrawing the improvement altogether (configurable).
- Transit improvements apply to the same prefixes as do outbound improvements. The outbound improvements are withdrawn in order to avoid the risk of contradictory routing decisions.

Implementing optimization of transiting traffic introduces a series of risks and some inherent drawbacks, for example:

- Potential blackholing of traffic when all alternative routes are withdrawn. IRP implements a protection against this by traversing all RIB-in entries for improved transit prefixes and confirming that the route is still being announced by other providers besides IRP. Still there can be a short time period between confirmations when all alternative routes have been withdrawn and IRP did not yet get a chance to re-confirm this. Attempting to reduce this time period by setting up a higher frequency of confirmations leads to increased load on the router and a tradeoff needs to be made for this. For example when the number of providers is quite large the probability that a route will be withdrawn through all of them is quite small and thus the frequency of confirmations can be reduced too.
- Additional CPU load on edge router(s) for servicing mandatory SNMP requests that check alternative route presence and BGP Best Path selection inside IRP.
- Working with missing BGP attributes that is not available over SNMP.
- Strict upper limits on the number of possible Inbound improvements are imposed by the trade-off required to reduce excessive router's CPU load.

Refer Optimization of transiting traffic for details.

1.2.23 Circuit issues detection

Starting with IRP 3.8 IRP adds excessive loss circuit issues detection features.

> Circuit issues detection feature is available for transit providers only.

When this feature is enabled for a provider IRP uses past probing data to detect when it suffers from excessive levels of packet loss. To determine excessive loss IRP compares a provider's average loss over an immediate past time horizon, number of probes and average loss difference from other providers. Depending on packet loss thresholds IRP can attempt different actions on the network.

Every time a circuit issue is detected IRP will raise corresponding alerts that can be subscribed to. Network engineers or external network management systems can act on them.

Additionally IRP even though it is constrained on how much can do, ensures that the following will take place:

- provider is marked with an issue badge and excluded from being considered a candidate for performance improvements,
- reprobing is performed for destination prefixes routed through affected provider,
- outbound improvements through affected provider are withdrawn,
- max prepends are announced for inbound and transit prefixes through affected provider,
- FlowSpec rules to induce drop of BGP session towards affected provider are added,

> Note that this is only possible if FlowSpec is enabled and the network is capable of processing FlowSpec rules. Dropping the BGP session with the affected provider causes all outbound and inbound traffic through this provider to be re-routed through known good providers.

- affected provider is monitored to detect if the issue was temporary and if loss averages return to normal restore it to a good state.

Enable this feature for each of the designated provider as detailed in Configuration editor: Provider name and review circuit issue detection thresholds as detailed in Core configuration.

1.2.24 Threat Mitigation

Starting with IRP 4.1, a new feature called Threat Mitigation is added to the platform. It incorporates the statistics collection as well as the blackholing mechanism, present in the previous product versions, nevertheless offering a fully automated threshold-based threat mitigation instrument that introduces Flowspec in addition to the RTBH.

RTBH feature allows redirection of traffic to a non-existent resource (a so-called black hole), or the blocking of the unwanted traffic in a provider's network to prevent such traffic from entering the user's network.

Threat Mitigation can operate in the following modes:

Automated IRP performs a particular threat mitigation action automatically when an attack is detected

Moderated stands for manual confirmation of the Threat Mitigation action. Once IRP detects an attack, a suggestion to enable the mitigation rule gets displayed in the GMI interface. IRP does not perform any actions without user confirmation. If the action is not enabled in time before a subsequent cycle of checks, IRP reviews the proposal and adjusts it as needed, keeping the recommendation, making a different one, or removing it altogether

Manual Allows to set up only manual threat mitigation rules

Disabled IRP does not perform any threat mitigation actions.

The Threat Mitigation feature is based on threshold rules set by IRP users. In the context of DoS/DDoS attack detection, threshold values refer to the rate of kilo packets or megabits per second. Specifically, a detection threshold represents the rate at which an IRP instance raises an alert for an attack and takes appropriate action as per the predefined rule. For instance, if the Flowspec rule threshold is set to 80 kilo packets per second, an alert will be generated once incoming packets flow towards a destination exceeds this bound and the consecutive Flowspec action gets triggered. Hence, indicating an appropriate detection threshold value in rules is critical to providing an efficacious response to DoS threats while reducing false alerts.

FlowSpec mitigation method can be chosen to filter out smaller attacks while the Remote Triggered Blackhole should be sent to providers to block large volume attacks.

1.2.24.1 Configuring Blackholing

A Provider in IRP should be configured before it could be used for blackholing.

IRP should know next-hop (bgpd.peer.X.blackholing.ipv4.next_hop, bgpd.peer.X.blackholing.ipv6.next_hop), localpref (bgpd.peer.X.blackholing.localpref) and community (peer.X.blackholing.community) values to be able to send a route to a user's network.

A user's router is responsible to distinguish communities sent by an IRP instance and advertise blackholing routes to a provider's router used to receive such routes.

1.2.24.2 Configuring FlowSpec

FlowSpec feature should be enabled globally in global.flowspec then for each BGP session in bgpd.peer.X.flowspec.

1.3 IRP Optimization modes

1.3.1 Performance optimization

Performance optimization mode makes sure that traffic flows through the best performing routes by reducing packet loss and latency, while ignoring other characteristics such as provider bandwidth usage or transit cost. The system analyzes each of the connected providers and compares only their performance metrics in order to choose the best candidate and make an improvement.

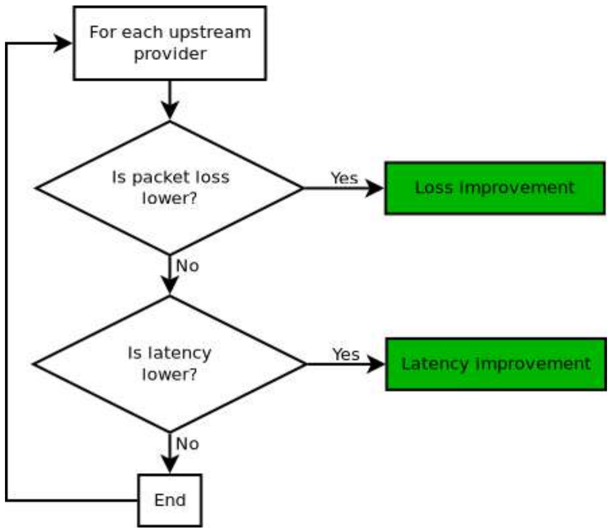

Figure 1.3.1: Performance optimization algorithm

First of all, IRP considers packet loss. If loss is lower and the difference is greater than a predefined value, then the system checks if sending the traffic through this provider will not cause any network congestion. If it confirms that the traffic can flow freely, the system declares the provider as the best one to route through.

However, if loss values are equal or the difference between providers is smaller than predefined, the system continues by comparing latency values. If latency is lower and the difference in latency between providers is greater than predefined, then the system declares a latency-based improvement.

1.3.2 Cost optimization

Cost optimization mode decreases packet loss and improves the cost while not worsening latency more than allowed. If IRP cannot improve costs it tries to reduce latency. The platform runs the same algorithm for loss comparison as in performance optimization mode.

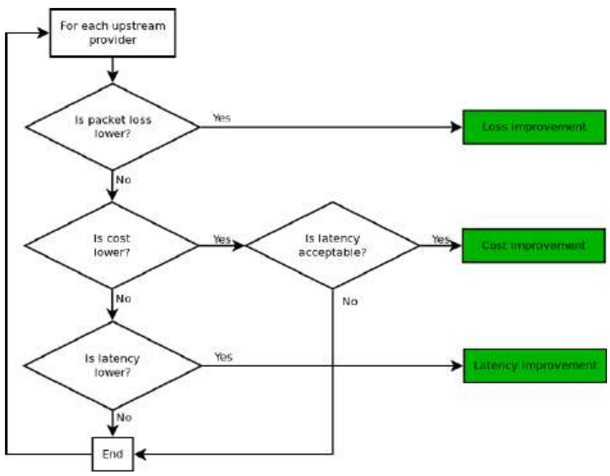

Figure 1.3.2: Cost optimization algorithm

Note that the diagram does not highlight higher cost cases. IRP operating in cost optimization mode can make improvements towards higher cost providers only when current routes suffer from loss.

Before comparing latency values IRP compares the transit cost for each provider. If the cost of the new provider is better, the system goes further by checking the latency and validates the cost improvement only if the latency is not worsening more than predefined. However, if the cost cannot be improved, IRP tries to make a latency improvement using the same algorithm as in performance mode. Thus, if there is no way to make a cost improvement, system reroutes the traffic to the best performing provider.

1.3.3 Commit Control

Commit Control, allows to keep the commit levels for each provider at a pre-configured level. It includes bandwidth control algorithms for each provider as well as the active traffic rerouting, in case bandwidth for a specific provider exceeds the configured limit. Commit Control also includes passive load adjustments inside each provider group.

A parameter called "precedence" (see peer.X.precedence) is used to set the traffic unloading priorities, depending on the configured bandwidth cost and providers throughput. The platform will reroute excessive bandwidth to providers, whose current load is less than their 95th percentile. If all providers are overloaded, traffic is rerouted to the provider with the smallest precedence - usually this provider has either the highest available bandwidth throughput, or the lowest cost. The higher is the precedence, the lower is the probability for the traffic to be sent to a provider, when its pre-configured 95th percentile usage is higher.

IRP usually allows CC improvements when the candidate providers have better or equal loss to current route. This can be configured under core.commit_control.loss_override.

See also: core.commit_control.

1.3.3.1 Flexible aggressiveness of Commit algorithm based on past overloads

Metering bandwidth usage by the 95th presents the following alternative interpretation - the customer is allowed to exceed his limits 5% of times. As such, IRP assumes there's a schedule of overloads based on the current time within a month and an actual number of overloads already made.

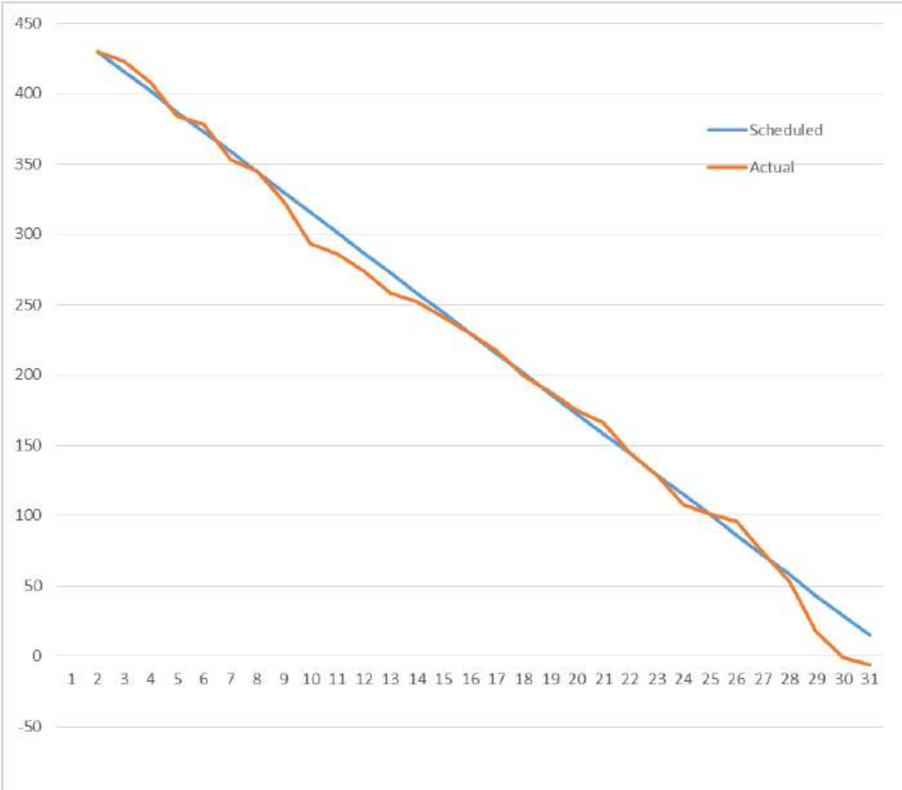

Figure 1.3.3: More aggressive when actual exceeds schedule

The sample image above highlights the remaining scheduled and the actual amount of allowed overloads for the month decreasing. Whenever IRP depicts that the actual line goes below schedule (meaning the number of actual overloads exceeds what is planned) it increases its aggressiveness by starting unloading traffic earlier than usual. For example, if the commit level is set at 1Gbps and IRP will start unloading traffic at possibly 90% or 80% depending on past overloads count.

The least aggressive level is set at 99% and the most aggressive level is constrained by configuration parameter core.commit_control.rate.low + 1%.

This is a permanent feature of IRP Commit Control algorithm.

1.3.3.2 Trigger commit improvements by collector

Commit control improvements are made after the flows carrying traffic are probed and IRP has fresh and relevant probe results. Subsequent decisions are made based on these results and this makes them more relevant. Still, probing takes some time and in case of fluctuating traffic patterns the improvements made will have reduced impact due to flows ending soon and being replaced by other flows that have not been probed or optimized.

For networks with very short flows (average flow duration under 5 minutes) probing represents a significant delay. In order to reduce the time to react to possible overload events IRP added the feature to trigger commit control improvements on collector events. When Flow Collector detects possible overload events for some providers, IRP will use data about past probed destinations in order to start unloading overloaded providers early. This data is incomplete and a bit outdated but still gives IRP the opportunity to reduce the wait time and prevent possible overloads. Later on, when probes are finished another round of improvements will be made if needed.

Due to the fact that the first round of improvements is based on older data, some of the improvements might become irrelevant very soon. This means that routes fluctuate more than necessary while on average getting a reduced benefit. This is the reason this feature is disabled by default. Enabling this feature represents a tradeoff that should be taken into consideration when weighing the benefits of a faster react time of Commit Control algorithm.

This feature is configurable via parameter: core.commit_control.react_on_collector. After enabling/disabling this feature IRP Core service requires restart.

1.3.3.3 Commit Control improvements on disable and re-enable

IRP up to version 2.2 preserved Commit Control improvements when the function was disabled globally or for a specific provider. The intent of this behavior was to reduce route fluctuation. In time these improvements are overwritten by new improvements.

We found out that the behavior described above ran contrary to customer's expectations and needs. Usually, when this feature is disabled it is done in order to address a more urgent and important need. Past Commit Control improvements were getting in the way of addressing this need and was causing confusion. IRP versions starting with 2.2 aligns this behavior with customer expectations:

- when Commit Control is disabled for a provider (peer.X.cc_disable = 1), this Provider's Commit Control improvements are deleted;
- when Commit Control is disabled globally (core.commit_control = 0), ALL Commit Control improvements are deleted.

1.3.3.4 Provider load balancing

Provider load balancing is a Commit Control related algorithm, that allows a network operator to evenly balance the traffic over multiple providers, or multiple links with the same provider.

For example, a specific network having an average bandwidth usage of 6Gbps has two separate ISPs. The network operator wants (for performance and/or cost reasons) to evenly push 3Gbps over each provider. In this case, both upstreams are grouped together (see peer.X.precedence), and the IRP system passively routes traffic for an even traffic distribution. Provider load balancing is enabled by default via parameter peer.X.group_loadbalance.

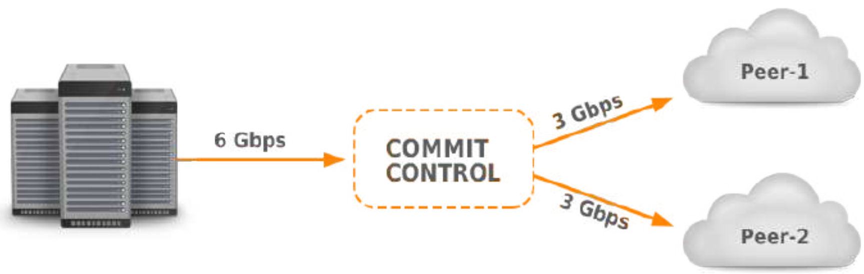

Figure 1.3.4: Provider load balancing

1.3.3.5 Commit control of aggregated groups

Customers can deploy network configurations with many actual links going to a single ISP. The additional links can serve various purposes such as to provision sufficient capacity in case of very large capacity requirements that cannot be fulfilled over a single link, to interconnect different points of presence on either customer (in a multiple routing domain configuration) or provider sides, or for redundancy purposes. Individually all these links are configured in IRP as separate providers. When the customer has an agreement with the ISP that imposes an overall limitation on bandwidth usage, these providers will be grouped together in IRP so that it can optimize the whole group.

The rationale of this feature as illustrated in the figure below is that if in the group overusages on one provider are compensated by underusages on another provider there is no need to take any action since overall the commitments made by the customer to the ISP have not been violated. Commit control algorithm will take action only when the sum of bandwidth usage on all providers in the group exceed the sum of bandwidth limits for the same group of providers.

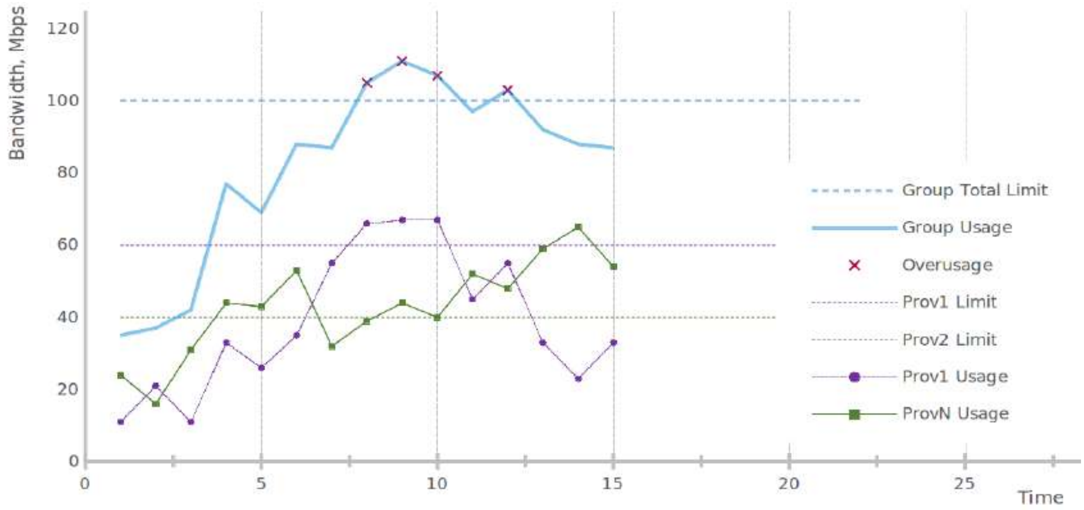

Figure 1.3.5: Commit control of aggregated groups

The image above highlights that many overusages on the green line are compensated by purple line underusages so that the group usage is below group total limits. Only when the traffic on the purple line increases significantly and there are no sufficient underusages on the other providers in the group to compensate the overusages, Commit Control identifies overusages (highlighted with a red x on the drawing above) and takes action by rerouting some traffic towards providers outside the group.

⚠ It is important to note that in order for this feature to be effective there must be providers configured in IRP that are not part of this group. This way when candidate improvements are considered there are alternative routes via those providers that the traffic can be rerouted to.

In order to configure providers that are optimized as an aggregated group 1) first the providers will be configured with the same precedence in order to form a group; 2) the overall 95th limitation will be distributed across providers in the group as appropriate; 3) and finally load balancing for the group will be Disabled by parameter peer.X.group_loadbalance.

1.3.3.6 95th calculation modes

Commit control uses the 95th centile to determine whether bandwidth is below or above commitments. There are different ways to account for Outbound and Inbound traffic when determining the 95th value. IRP supports the following 95th calculation modes:

- Separate 95th for in/out: The 95th value for inbound and outbound traffic are independent and consequently bandwidth control for each is performed independently of each other. For this 95th calculation modes IRP monitors two different 95th for each inbound and outbound traffic levels.
- 95th from greater of in, out: At each time-point the greater of inbound or outbound bandwidth usage value is used to determine 95th.
- Greater of separate in/out 95th: 95th are determined separately for inbound and outbound traffic and the larger value is used to verify if commitments have been met.

Refer 4.14.5.

1.3.3.7 Other commit control features

Current improvements report has been improved to keep CC improvements applied during an algorithm cycle in a single sorted batch. This way the report no longer confuses by intermingling the improvements.

7	☐	14:28:59	18.159.0.0/16	16509	Amazon.com, Inc.	0 → 0	41 → 37	Orange	StarNet		global (-)
8	☐	14:25:28	73.85.0.0/16	7922	Comcast Cable Communications, LLC	0 → 0	165 → 165	Orange	StarNet		global (-)
9	☐	13:58:35	178.175.128.0/20	43289	Trabia SRL	0 → 0	1 → 12	MDIX	StarNet		global (-)
10	☐	13:53:57	52.51.0.0/16	16509	Amazon.com, Inc.	0 → 0	64 → 60	Orange	StarNet		global (-)

Figure 1.3.6: Sorted Commit Control improvements

The excerpt above highlights the new commit control improvements done at the same time. They unload 9.11, 6.09, 5.70, 4.26 Mbps from provider B to A thus unloading B from 109% to 58% and increasing the load on A from 11% to 46%. The data would have been more confusing if the improvements in that single batch were intermingled.

During retry probing IRP will delete Commit Control improvements with current bandwidth less than the configured threshold given by core.commit_control.agg_bw_min.

1.4 What is IRP Global Management Interface

IRP Global Management Interface (GMI) is a pane of glass interface that allows administrators to manage multiple Noction Intelligent Routing Platform instances and get access to various data and statistics for those instances from one, easy to access application.

GMI allows administrators to:

- Monitor multiple IRP instances and their performance
- Get comprehensive network performance analytics
- Facilitate network troubleshooting
- Automatically manage bandwidth levels for provider groups from various points of presence using the Global Commit feature

1.4.1 GMI Operating Details

GMI act as a standard web application, all user interactions are made via a web browser. The list of supported browsers includes:

- Google Chrome / Chromium version 42 and higher
- Safari version 10.1 and higher
- Firefox 64 and higher
- Edge 45 and higher
- Opera 42 and higher

1.4.2 GMI Technical Requirements

To deploy GMI, a series of hardware/software requirements need to be met

1.4.2.1 Hardware requirements

1. CPU
    - Recommended Intel® i5 latest 10th, 9th, 8th, 7th, and 6th Generation Products with 5 threads
2. RAM

- Recommended size of at least 4 GB

3. HDD

   - Recommended size of at least 4 GB SSD

1.4.2.2 Software requirements

Clean Linux system, with the latest Red Hat Enterprise Linux 8, Red Hat Enterprise Linux 9 (including binary compatible systems) or Ubuntu Server LTS of x86_64 architecture installed on the server.

Chapter 2

Configuration

2.1 IRP software management

Software repository

IRP packages (current, new and old versions) are available in Noction's repositories hosted at repo.noction.com. Coordinate getting access to the repository with Noction's representatives.

Software installation and upgrade

Once the IRP packages repository is configured standard OS package management tools (dnf/apt) are used to install, upgrade or downgrade IRP.

Installation:

Red Hat Enterprise Linux 8 and 9:

```
dnf config-manager --enable repo PowerTools
dnf install irp
```

Ubuntu Server:

```
apt-get update
apt-get install irp
```

Optional package *irp-documentation* containing this documentation in PDF format and API reference should be installed separately.

Upgrade to latest version:

Red Hat Enterprise Linux 8 and 9:

```
dnf upgrade "irp*"
```

Ubuntu:

```
apt-get update
apt-get install --only-upgrade irp\*
```

Downgrade to a specific version:

Red Hat Enterprise Linux 8 and 9:

```
dnf downgrade "irp*3.7*"
```

Ubuntu:

> IRP added support for Ubuntu Server starting with version 3.9. IRP versions prior to 3.9 will not be offered for Ubuntu Server.

Check for available IRP versions:

```
apt-cache policy irp
```

Edit file /etc/apt/preferences.d/irp (refer to apt_preferences man page for package pinning) to pin desired version:

```
Package: irp irp-*
Pin: version 3.9.0-RELEASE~build11806~trusty
Pin-Priority: 1001
```

```
apt-get update
apt-get upgrade irp
```

2.2 Configuration files

IRP is currently configured using a set of textual Unix-style configuration files, located under the `/etc/noction/` directory. In-line comments can be added to any line, using the "#" symbol.

Several separate configuration files are presented as follows:

- `/etc/noction/irp.conf` - the main IRP configuration file contains configuration parameters for all IRP components, including algorithm parameters, optimization modes definitions and providers settings.
- `/etc/noction/db.global.conf` - database configuration file for all IRP components.
- `/etc/noction/exchanges.conf` - Exchanges configuration file
- `/etc/noction/inbound.conf` - Inbound prefixes configuration file (3.12.5.1)
- `/etc/noction/policies.conf` - Routing Policies configuration file (1.2.9).

Additional configuration files can be used for several core, global and explorer preferences, as described in sections 4.2.24, 4.7.24 and 4.9.4.

A comprehensive list of all the IRP parameters along with their description can be found in the Configuration parameters reference chapter.

2.3 IRP Global Management Interface Configuration

2.3.1 GMI Initial Configuration

After the installation, GMI service will start automatically. On startup, GMI service starts listening on the following network ports: 80, 443.

When installed on the same server with an IRP instance, GMI service listens on the following network ports: 1443, 1080.

The GMI initial configuration can be performed by using the Initial Setup wizard (Initial Setup) which can be accessed via the main platform IP address over HTTP or HTTPS protocol.

2.3.2 GMI software management

Software repository

GMI packages (current and new) are available in Noction's repositories hosted at repo.noction.com. Coordinate getting access to the repository with Noction's representatives.

Software installation and upgrade

Once the GMI packages repository is configured, follow the instructions below to install or upgrade GMI.

Installation

RedHat Enterprise Linux 8 and later

```
dnf install irp-gmi
```

Ubuntu

```
apt-get update
apt-get install irp-gmi
```

Upgrade to the latest version

RedHat Enterprise Linux 8 and later

```
dnf upgrade irp-gmi
```

Ubuntu

```
apt-get update
apt-get upgrade
```

2.3.3 Starting, stopping and getting status of GMI backend

GMI acts as a Linux service, and can be started, stopped or restarted from the Linux terminal, using the commands listed below:

⚠ Critical errors preventing startup of a particular component are logged to journalctl. Details can be obtained from logs stored in journalctl using the following command

```
journalctl -fu irp-gmi
```

Starting backend:

```
systemctl start irp-gmi
```

Stopping backend:

```
systemctl stop irp-gmi
```

Restarting backend:

```
systemctl restart irp-gmi
```

Backend status:

```
systemctl status irp-gmi
```

2.4 Starting, stopping and getting status of IRP components

In order to determine the status of IRP components or to start, stop and restart them standard Linux utilities are used.

⚠ Critical errors preventing startup of a particular component are logged to console. Details can be obtained from logs stored in */var/log/irp/* directory.

Managing software components in OS with systemd

Starting single component:

```
systemctl start explorer
```

ℹ Multiple components separated by space can be listed

Stopping single component:

```
systemctl stop explorer
```

ℹ Multiple components separated by space can be listed

Starting all components:

```
systemctl start irp.target
```

ℹ This also starts all prerequisites

Stopping all components:

```
systemctl start irp-shutdown.target
```

Stopping all components except bgpd:

```
systemctl start irp-shutdown-except-bgpd.target
```

Restarting all components:

```
systemctl start irp-shutdown.target
systemctl start irp.target
```

Obtaining overall status of all components:

```
systemctl list-dependencies irp.target
```

ℹ Individual statuses can be checked by executing command *systemctl status component_name*

2.5 IRP Initial Configuration

The platform initial configuration can be performed by using the Initial Setup wizard (Initial Setup) which can be accessed via the main platform IP address over HTTP or HTTPS protocol.

Example: https://10.11.12.13

2.6 Global and Core Configuration

The default values, specified for the Core service are sufficient for a proper system start-up. Several parameters can be adjusted during the initial system deployment. For a comprehensive list please see the Global parameters and Core settings sections.

During the initial setup and configuration stage, one must pay attention to the following configuration parameters:

- global.nonintrusive_bgp - must be set to “1” until the configuration and route propagation tests are completed.
- global.improve_mode - must be configured according to the specific network operator policies. See also: IRP Optimization modes
- global.aggregate - in most cases, it is recommended to enable the aggregates, in order to reduce the number of prefixes advertised by the IRP. Please consult the network infrastructure and configuration.
- core.commit_control - should be configured according to the specific network operator policies, see also: Commit Control
- core.outage_detection - in most cases it must be enabled. For more details see Outage detection

2.7 Collector Configuration

Depending on the preferred method of traffic data collection, one or both collector components should be configured. As specified in the IRP Components section, IRP can gather traffic data using the Flow (from now on: `irpflowd`) and Span collector (`irpspand`).

First of all, specific configuration to each collector will be described, along with the required router configuration changes.

2.7.1 Irpflowd Configuration

`Irpflowd` is a NetFlow/sFlow collector that receives and analyzes network traffic information, generated by your router(s).

NetFlow - an IP network statistics protocol developed by Cisco Systems, Inc. for collecting and transferring statistics regarding IP traffic information from network devices such as switches/routers to network analysis applications. `Irpflowd` currently supports the following **NetFlow** versions: v1, v5, v9.

sFlow - is a protocol designed for monitoring network, wireless and host devices. Developed by the sFlow.org Consortium, it is supported by a wide range of network devices, as well as software routing and network solutions.

> Flow collector use is mandatory for Multiple Routing Domain networks since SPAN does not carry attributes to distinguish traffic between different providers.

2.7.1.1 Flow agents

Multi-router networks usually simultaneously carry traffic to a prefix over multiple providers. Flow collector needs to know the exact details of such a configuration in order to correctly determine the overall provider volume and active flows. Each provider configured in IRP can be explicitly set to match Flow statistics to specific Flow agents and help IRP Flow collector assign accurate statistics for each provider.

Flow agents have been added in order to support Optimization for multiple Routing Domains but when available they are used to enhance IRP capabilities in other areas too. For example a correct set of Flow agents for all providers enables IRP to accurately determine a prefix's current route. IRP components, especially Core during decision making can spot the latest prefix statistics matching a given Flow agent(s) and infer amount of traffic is sent over individual Providers. Collected data is later used to make Performance and Bandwidth decisions with knowing about multiple best routes.

> In case Flow agent data is missing IRP relies on past probing data to determine the current route of a prefix.

Flow agents are specified in the form of:

```
IPv4/interfaceID
```

where IPv4 is the source IP address of the packets coming from the agent and interfaceID is a numerical identifier of the interface in the range 1- 4294967295. The interface ID is usually its SNMP Interface ID on the router.

A collection of such values is assigned when multiple physical interfaces are used. For example:

```
peer.X.flow_agents = 8.8.8.8/1 8.8.8.8/2 8.8.8.8/3 8.8.8.8/4
```

The value is set via parameter peer.X.flow_agents or under Providers and Peers configuration in Frontend. The Frontend will also retrieve and suggest a list of available values that can be matched with the provider.

2.7.1.2 TCP port collection for outbound IPs

IRP's can monitor TCP ports used by outbound IP addresses to identify open service ports and evaluate network performance using TCP CONNECT probes. This feature is configurable and provides flexibility in how many ports can be collected.

The main control for this functionality is the collector.flow.tcp_ports.mode parameter, which defines the port collection method. Depending on the selected mode, the system can either collect ports within a defined range, gather only those listed explicitly, select the lesser port numbers of the one chosen by inbound IP address, or skip port collection entirely. By default, port collection is disabled.

To manage the volume of data, the collector.flow.tcp_ports.limit parameter sets a cap on the number of TCP ports collected per outbound IP address during a one-minute interval. This limit also applies to the Explorer component when determining how many ports to probe for a given IP. The default limit is 5, and values between 1 and 50 are allowed.

When using the range-based collection mode, the start and end of the monitored port range are configured with collector.flow.tcp_ports.min and collector.flow.tcp_ports.max, respectively. By default, the Irpflowd monitors ports from 1 to 3000.

Alternatively, a specific list of TCP ports can be defined using collector.flow.tcp_ports.list, which becomes active when the collection mode is set to "Collect ports in list".

This functionality ensures that IRP targets relevant and reachable TCP ports when collecting data and performing active measurements, while maintaining efficient data handling through customizable limits and flexible collection scopes.

2.7.1.3 Configuration

To use the `irpflowd` collector, the following steps must be completed:

1. NetFlow/sFlow/jFlow must be configured on the router(s), which must send traffic flow information to the main IRP server IP (Figure 2.7.1). See (2.7.1.4) for specific network device configuration instructions

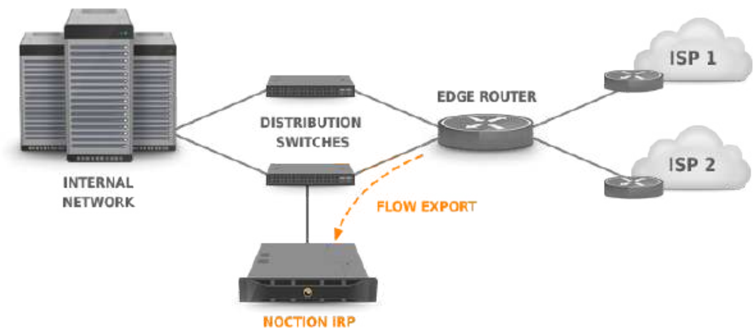

Figure 2.7.1: Flow export configuration

2. Irpflowd must be enabled (by setting the collector.flow.enabled parameter):

```
collector.flow.enabled = 1
```

3. A list of all the networks, advertised by the edge routers that IRP will optimize, should be added to the configuration. This information should be specified in the collector.ournets parameter.

4. For security reasons, the list of valid Flow sending IP addresses must be configured in the collector.flow.sources, to protect `irpflowd` from unauthorized devices sending Flow data.
Example:

```
collector.flow.sources = 10.0.0.0/29
```

5. In case the Flow exporters are configured to use non-standard port numbers (`2055` for NetFlow/jFlow and `6343` for sFlow), then collector.flow.listen.nf and collector.flow.listen.sf must be adjusted accordingly:

```
collector.flow.listen.nf = 2055
collector.flow.listen.sf = 6343
```

2.7.1.4 Vendor-specific NetFlow configuration examples

The following sections contain vendor-specific configuration examples, along with possible pitfalls and version-specific issues

NetFlow configuration on Cisco 7600/6500 series routers

Refer also to the Cisco NetFlow Software Configuration Guide

Listing 2.1: Global MLS settings configuration

```
(config)# mls netflow
(config)# mls flow ip interface-full
(config)# mls flow ipv6 interface-full
(config)# mls sampling packet-based 512 8192
(config)# mls nde sender version 7
```

Please replace the IP address and port with the actual IRP host IP and the collector UDP port (`2055` by default)

Listing 2.2: Global NetFlow settings and export configuration

```
(config)# ip flow-cache entries 524288
(config)# ip flow-cache timeout inactive 60
(config)# ip flow-cache timeout active 1
(config)# ip flow-export version 9
(config)# ip flow-export destination 10.11.12.14 2055
```

Ingress flow collection must be enabled on all interfaces facing the internal network.
MLS NetFlow sampling must be enabled to preserve router resources.

Listing 2.3: Per-interface NetFlow settings configuration

```
(config)# int GigabitEthernet 3/6
(config-if)# mls netflow sampling
(config-if)# ip flow ingress
```

Flexible NetFlow configuration on Cisco 6500 series routers IOS 15.0SY series

Listing 2.4: Flexible NetFlow monitor configuration

```
(config)# flow monitor IRP-FLOW-MONITOR
(config-flow-monitor)# record platform-original ipv4 full
(config-flow-monitor)# exporter IRP-FLOW-EXPORTER
(config-flow-monitor)# cache timeout inactive 60
(config-flow-monitor)# cache timeout active 60
(config-flow-monitor)# cache entries 1048576
```

Listing 2.5: Flexible NetFlow exporter configuration

```
(config)# flow exporter IRP-FLOW-EXPORTER
(config-flow-exporter)# destination 10.11.12.14
(config-flow-exporter)# source Loopback0
(config-flow-exporter)# transport udp 2055
(config-flow-exporter)# template data timeout 120
```

Please replace the IP address and port with the actual IRP host IP and the collector UDP port (`2055` by default). Also replace the source interface with the actual one.

Listing 2.6: Flexible NetFlow sampler configuration

```
(config)# sampler flow-sampler
(config-sampler)# mode random 1 out-of 1024
```

Listing 2.7: Per-interface Flexible NetFlow settings configuration

```
(config)# interface FastEthernet0/0
(config-if)# ip flow monitor IRP-FLOW-MONITOR sampler flow-sampler input
(config-if)# ip flow monitor IRP-FLOW-MONITOR sampler flow-sampler output
```

NetFlow configuration on Cisco 7200/3600 series routers

Please replace the IP address and port with the actual IRP host IP and the collector UDP port (`2055` by default)

Do not attempt to configure 7600/6500 series routers according to 7200/3600 router's configuration guide.

Listing 2.8: NetFlow configuration on Cisco 7200/3600 series routers

```
Router(config)# ip flow-cache entries 524288
Router(config)# ip flow-cache timeout inactive 60
Router(config)# ip flow-cache timeout active 1
Router(config)# ip flow-export version 9
Router(config)# ip flow-export destination 10.11.12.14 2055
```

Ingress/egress flow export configuration on peering interfaces

According to Cisco IOS NetFlow Command Reference regarding the "ip flow" command history in IOS releases, this feature was introduced in IOS 12.3(11)T, 12.2(31)SB2, 12.2(18)SXE, 12.2(33)SRA.

NetFlow exporting must be configured on each peering interface which is used to send and/or receive traffic:

Listing 2.9: Ingress/egress flow export configuration on peering interfaces

```
Router(config)#interface FastEthernet 1/0
Router(config-if)#ip flow ingress
Router(config-if)#ip flow egress
```

Ingress flow export configuration (earlier IOS releases)

Ingress flow export must be enabled on all interfaces facing the internal network. Prior to IOS 12.3(11)T, 12.2(31)SB2, 12.2(18)SXE, 12.2(33)SRA, flow export can be enabled only for ingress traffic, therefore it must be enabled on each interface that transmits/receives traffic from/to networks that must be improved

Listing 2.10: Ingress flow export configuration

```
Router(config)#interface FastEthernet 1/0
Router(config-if)#ip route-cache flow
```

NetFlow/sFlow configuration examples for Vyatta routers (VC 6.3)

Do not configure both NetFlow and sFlow export for the same router interface. It will lead to traffic statistics distortion.

Sampling interval must be set to at least 2000 for 10G links and 1000 for 1G links in order to save resources.

Listing 2.11: Configuring NetFlow export on Vyatta

```
vyatta@vyatta# set system flow-accounting netflow server 10.11.12.14 port
    2055
vyatta@vyatta# set system flow-accounting netflow version 5
```

Listing 2.12: Configuration of an interface for the flow accounting

```
vyatta@vyatta# set system flow-accounting interface eth0
vyatta@vyatta# commit
```

jFlow export configuration for Juniper routers

Routing Engine-based sampling supports up to eight flow servers for both version 5 and version 8 configurations. The total number of collectors is limited to eight, regardless of how many are configured for version 5 or version 8. During the sampling configuration, the export packets are replicated to all the collectors, configured to receive them. If two collectors are configured to receive version 5 records, then both collectors will receive records for a specified flow.

Default export interval for active/inactive flows on some Juniper routers is 1800 seconds. IRP requires significantly more frequent updates. The export interval recommended by IRP is 60 seconds. Refer your JunOS documentation on how to set export intervals. These parameters are named flow-active-timeout and flow-inactive-timeout.

Listing 2.13: Juniper flow export configuration

```
forwarding-options {
      sampling {
            input {
                  family inet {
                        rate 1000;
                  }
            }
      }
      family inet {
            output {
```

```
                        flow-server 10.10.3.2 {
                                port 2055;
                                version 5;
                                source-address 10.255.255.1;
                        }
                }
        }
}
```

NetFlow export must be configured on all the interfaces facing the providers. In some cases, it may also be necessary to enable NetFlow forwarding on the interfaces facing the internal network.

Listing 2.14: Per-interface NetFlow sampling configuration

```
interfaces {
        xe-0/0/0 {
                unit 0 {
                        family inet {
                                sampling {
                                        input output;
                                }
                        }
                }
        }
}
```

2.7.2 Irpspand Configuration

`Irpspand` acts like a passive network sniffer, analyzing the traffic that is provided to one or more dedicated network interfaces on the IRP server (defined in collector.span.interfaces) from a mirrored port on your router or switch. Irpspand looks up the IP header in the mirrored traffic. When the link level traffic is VLAN tagged as for example in IEEE 802.1Q or 802.1ad for Q-in-Q datagrams, Irpspand will advance its IP packet sniffer past VLAN tags. For better results (higher performance and more analyzed traffic), as specified in the IRP Technical Requirements section, we recommend using Myricom 10Gbps NICs, with Sniffer10G license enabled.

⚠ Enable collector.span.size_from_ip_header configuration parameter if packets are stripped before forwarding to IRP's SPAN port.

To use the `irpspand` collector, the following steps must be completed:

1. Configure port mirroring on your router or switch, as shown in figures (2.7.2) and (2.7.3).

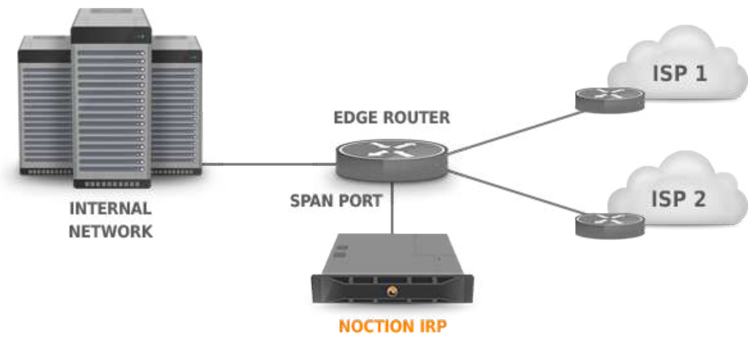

Figure 2.7.2: Span port configuration (on the router)

or:

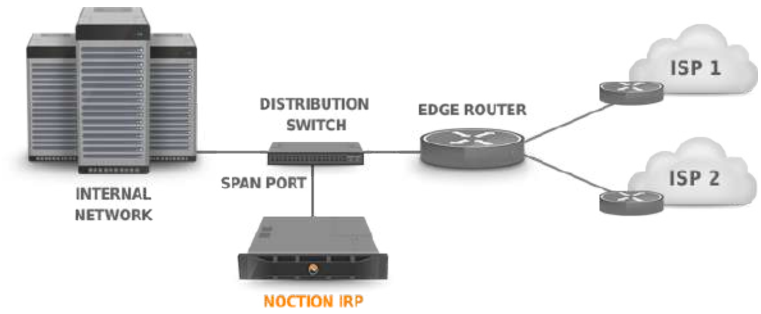

Figure 2.7.3: Span port configuration (on the switch)

2. Enable the span collector by setting the collector.span.enabled parameter in the configuration file:

```
collector.span.enabled = 1
```

3. Define the list of network interfaces that receive mirrored traffic, by setting the collector.span.interfaces parameter (multiple interfaces separated by space can be specified):

```
collector.span.interfaces = eth1 eth2 eth3
```

4. A list of all the networks advertised by the edge routers that IRP will optimize must be added to the configuration. This information should be specified in the collector.ournets parameter.

5. In case blackouts, congestions and excessive delays are to be analyzed by the system, the collector.span.min_delay must be turned on as well

```
collector.span.min_delay = 1
```

2.8 Explorer Configuration

As described in the IRP Components section, the explorer is the actual service that performs active remote network probing and tracing.

Explorer runs active and passive probings through all the available providers in order to determine the best one for the particular prefix. Such metrics as packet loss, latency, link capacity and link load are taken into consideration. In order to run a probe through a particular provider, Explorer needs the following to be configured:

1. An additional IP alias for each provider should be assigned and configured on the IRP server. This IP will be used as a source address during the probing process.

> It is recommended to configure reverse DNS records for each IP using the following template: `performance-check-via-<PROVIDER-NAME>.HARMLESS-NOCTION-IRP-PROBING.<YOUR-DOMAIN-NAME>.`

2. Policy-based routing (PBR) has to be configured on the edge router(s), so that traffic originating from each of these probing IP addresses will exit the network via specific provider. See specific PBR configuration in the Specific PBR configuration scenarios section.

> If network has Flowspec capabilities then alternatively Flowspec policies can be used instead of PBR. Refer for example Flowspec policies, global.flowspec.pbr.

1. Policy-based routing has to be configured to drop packets rather than routing them through the default route in case that the corresponding Next-Hop does not exist in the routing table.

2.8.1 Specific PBR configuration scenarios

PBRs can be setup in multiple ways, depending on the existing network infrastructure.

We will assume the following IP addresses/networks are used:

10.0.0.0/24 - used on the IRP server as well as the probing VLANs

- **10.0.0.2/32** - main IRP server IP address
- **10.0.0.3-10.0.0.5** - probing IP addresses
- **10.0.0.250-10.0.0.254** - router-side IP addresses for the probing VLANs
- **10.0.1.0/24** - used for GRE tunnel interfaces, if needed
- **10.10.0.0/24** - real edge routers IP addresses

10.11.0.0/30 - BGP session with the 1st provider, **10.11.0.1** being the ISP BGP neighbor IP

10.12.0.0/30 - BGP session with the 2nd provider, **10.12.0.1** being the ISP BGP neighbor IP

10.13.0.0/30 - BGP session with the 3rd provider, **10.13.0.1** being the ISP BGP neighbor IP

Vlan 3 - the probing Vlan

eth0 - the probing network interface on the IRP server

> In a production network, please change the IP addresses used in these examples to the actual addresses assigned and used in the network. Same goes for the Vlan interface.

Brocade routers use Cisco compliant configuration commands. Unless otherwise noted, the Cisco configuration examples will also work on Brocade devices.

Case 1: Single router, two providers, and separate probing Vlan.

10.0.0.1/32 is configured on the edge router, on the probing vlan interface (ve3).

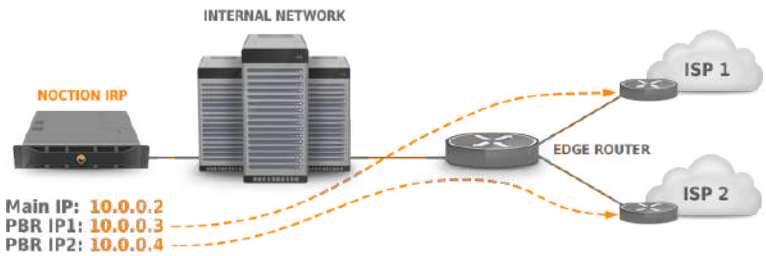

Figure 2.8.1: PBR configuration: single router and separate probing Vlan

In this case, a single PBR rule should be enforced on the Edge router for each of the probing IP addresses.

Listing 2.15: Cisco IPv4 PBR configuration: 1 router, 2 providers

```
access-list 1 permit ip host 10.0.0.3
access-list 2 permit ip host 10.0.0.4
!
route-map irp-peer permit 10
 match ip address 1
 set ip next-hop 10.11.0.1
 set interface Null0
!
route-map irp-peer permit 20
 match ip address 2
 set ip next-hop 10.12.0.1
 set interface Null0
!
interface ve 3
ip policy route-map irp-peer
```

Route-map entries with the destination set to Null0 interface are used for preventing packets to flow through the default route in the case that the corresponding next-hop does not exist in the routing table (typically when a physical interface administrative/operational status is down).

Cisco ASR9000 routers running IOS XR use a different PBR syntax.

Listing 2.16: Cisco IPv4 PBR configuration for IOS XR: 1 router, 2 providers

```
 configure
   ipv4 access-list irp-peer
     10 permit ipv4 host 10.0.0.3 any nexthop1 ipv4 10.11.0.1 nexthop2 ipv4
          169.254.0.254
     11 permit ipv4 host 10.0.0.4 any nexthop1 ipv4 10.12.0.1 nexthop2 ipv4
          169.254.0.254
   end
```

```
router static
 address-family ipv4 unicast
 169.254.0.254 Null0
 end

interface FastEthernet1/1
 ipv4 access-group irp-peer ingress
 end
```

For Juniper routers, a more complex configuration is required:

Listing 2.17: Juniper IPv4 PBR configuration: 1 router, 2 providers

```
[edit interfaces]
xe-0/0/0 {
      unit 3 {
         family inet {
            filter {
                  input IRP-policy;
            }
            }
      }
}
[edit firewall]
family inet {
  filter IRP-policy {
     term irp-peer1 {
        from {
           source-address 10.0.0.3/32;
        }
        then {
           routing-instance irp-isp1-route;
        }
      }
      term irp-peer2 {
         from {
            source-address 10.0.0.4/32;
         }
         then {
            routing-instance irp-isp2-route;
         }
      }
      term default {
         then {
            accept;
         }
      }
   }
}
[edit]
routing-instances {
   irp-isp1-route {
      instance-type forwarding;
      routing-options {
         static {
            route 0.0.0.0/0 next-hop 10.11.0.1;
         }
      }
   }
```

```
   irp-isp2-route {
      instance-type forwarding;
      routing-options {
         static {
            route 0.0.0.0/0 next-hop 10.12.0.1;
         }
      }
   }
}
routing-options {
   interface-routes {
      rib-group inet irp-policies;
   }
   rib-groups {
      irp-policies {
         import-rib [ inet.0 irp-isp1-route.inet.0 irp-isp2-route.inet.0 ];
      }
   }
}
```

PBR configuration on Vyatta routers.

Unfortunately, prior to and including VC6.4, Vyatta does not natively support policy-based routing. Thus, the PBR rules should be configured using the standard Linux `ip` toolset. To make these rules persistent, they should be also added to `/etc/rc.local` on the Vyatta system.

Listing 2.18: Vyatta IPv4 PBR configuration example

```
ip route add default via 10.11.0.1 table 101
ip route add default via 10.12.0.1 table 102
ip rule add from 10.0.0.3 table 101 pref 32001
ip rule add from 10.0.0.4 table 102 pref 32002
```

Vyatta versions VC6.5 and up, natively support source-based routing. The following example can be used:

Listing 2.19: Vyatta (VC6.5 and up) IPv4 PBR configuration example

```
# Setup the routing policy:
set policy route IRP-ROUTE
set policy route IRP-ROUTE rule 10 destination address 0.0.0.0/0
set policy route IRP-ROUTE rule 10 source address 10.0.0.3/32
set policy route IRP-ROUTE rule 10 set table 103
set policy route IRP-ROUTE rule 20 destination address 0.0.0.0/0
set policy route IRP-ROUTE rule 20 source address 10.0.0.4/32
set policy route IRP-ROUTE rule 20 set table 104
set policy route IRP-ROUTE rule 30 destination address 0.0.0.0/0
set policy route IRP-ROUTE rule 30 source address 0.0.0.0/0
set policy route IRP-ROUTE rule 30 set table main
commit

# Create static route tables:
set protocols static table 103 route 0.0.0.0/0 nexthop 10.11.0.1
set protocols static table 104 route 0.0.0.0/0 nexthop 10.12.0.1
commit

# Assign policies to specific interfaces, Vlan 3 on eth1 in this example:
set interfaces ethernet eth1.3 policy route IRP-ROUTE

# Verify the configuration:
show policy route IRP-ROUTE
```

```
show protocols static
show interfaces ethernet eth1.3
```

Case 2: Two edge routers, two providers, and a separate probing Vlan.

Following IP addresses are configured on the routers:
- `10.0.0.251` is configured on **R1, VE3**
- `10.0.0.252` is configured on **R2, VE3**

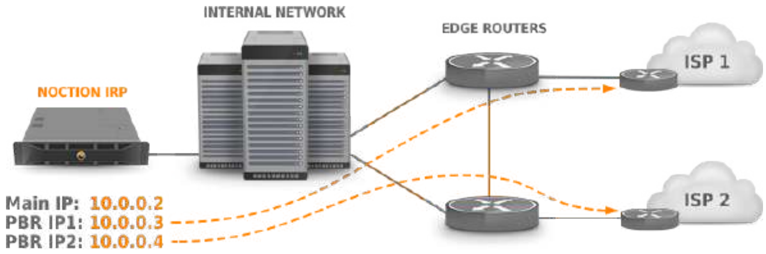

Figure 2.8.2: PBR configuration: two routers and separate probing Vlan

To reduce the number of PBR rules to one per each router, additional source based routing rules must be configured on the IRP server.

Listing 2.20: IRP server source-based routing using the 'ip' command

```
ip route add default via 10.0.0.251 table 201
ip route add default via 10.0.0.252 table 202
ip rule add from 10.0.0.3 table 201 pref 32101
ip rule add from 10.0.0.4 table 202 pref 32102
```

Refer to OS configuration manual for configuration guidelines.

Router configuration looks similar to the previous case. A Cisco/Brocade example will be provided.

Some Brocade routers/switches have PBR configuration limitations. Please refer to the "Policy-Based Routing" → "Configuration considerations" section in the Brocade documentation for your router/switch model.

For example, BigIron RX Series of switches do not support more than 6 instances of a route map, more than 6 ACLs in a matching policy of each route map instance, and more than 6 next hops in a set policy of each route map instance.

On the other hand, some Brocade CER/CES routers/switches have these limits raised up to 200 instances (depending on package version).

Listing 2.21: Cisco IPv4 PBR configuration: 2 routers, 2 providers, separate Vlan

```
#Router R1
access-list 1 permit ip host 10.0.0.3
!
route-map irp-peer permit 10
 match ip address 1
 set ip next-hop 10.11.0.1
 set interface Null0
!
```

```
interface ve 3
ip policy route-map irp-peer

#Router R2
access-list 1 permit ip host 10.0.0.4
!
route-map irp-peer permit 10
 match ip address 1
 set ip next-hop 10.12.0.1
 set interface Null0
!
interface ve 3
ip policy route-map irp-peer
```

Case 3: Complex network infrastructure, multiple routers, no probing VLAN

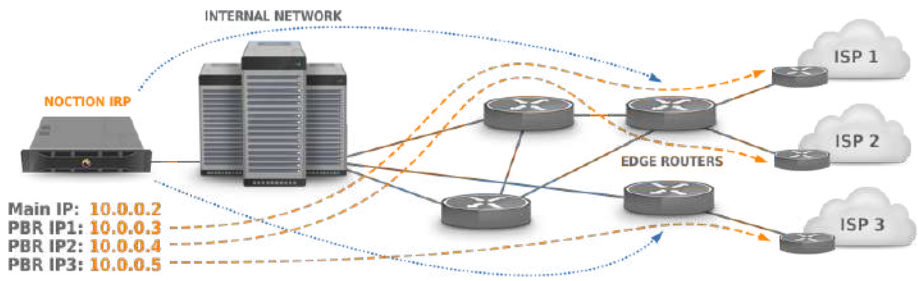

Figure 2.8.3: PBR configuration via GRE tunnels

In specific complex scenarios, traffic from the IRP server should pass multiple routers before getting to the provider. If a separate probing Vlan cannot be configured across all routers, GRE tunnels from IRP to the Edge routers should be configured.

> It is also possible to configure the PBRs without GRE tunnels, by setting PBR rules on each transit router on the IRP↔ provider path.

> Brocade routers do not support PBR set up on GRE tunnel interfaces. In this case the workaround is to configure PBR on each transit interface towards the exit edge router(s) interface(s).

GRE Tunnels configuration

Listing 2.22: IRP-side IPv4 GRE tunnel CLI configuration

```
modprobe ip_gre
ip tunnel add tun0 mode gre remote 10.10.0.1 local 10.0.0.2 ttl 64 dev eth0
ip addr add dev tun0 10.0.1.2/32 peer 10.0.1.1/32
ip link set dev tun0 up
```

Listing 2.23: IRP-side IPv4 GRE tunnel configuration using standard CentOS configs

```
#/etc/sysconfig/network-scripts/ifcfg-tun0
DEVICE=tun0
TYPE=GRE
ONBOOT=yes
MY_INNER_IPADDR=10.0.1.2
MY_OUTER_IPADDR=10.0.0.2
PEER_INNER_IPADDR=10.0.1.1
PEER_OUTER_IPADDR=10.10.0.1
TTL=64
```

Refer to OS configuration manual for configuration guidelines.

Listing 2.24: Router-side IPv4 GRE tunnel configuration (Vyatta equipment)

```
set interfaces tunnel tun0
set interfaces tunnel tun0 address 10.0.1.1/30
set interfaces tunnel tun0 description "IRP␣Tunnel␣1"
set interfaces tunnel tun0 encapsulation gre
set interfaces tunnel tun0 local-ip 10.10.0.1
set interfaces tunnel tun0 remote-ip 10.0.0.2
```

Listing 2.25: Router-side IPv4 GRE tunnel configuration (Cisco equipment)

```
interface Tunnel0 routers
ip address 10.0.1.1 255.255.255.252
tunnel mode gre ip
tunnel source Loopback1
tunnel destination 10.0.0.2
```

Listing 2.26: Router-side IPv4 GRE tunnel configuration (Juniper equipment)

```
interfaces {
     gr-0/0/0 {
         unit 0 {
             tunnel {
                 source 10.0.0.2;
                 destination 10.10.0.1;
             }
             family inet {
                 address 10.0.1.1/32;
             }
         }
     }
 }
```

The above configuration is for the first edge router (R1). One more GRE tunnel should be configured on the 2nd router (R2).

As soon as the GRE tunnels are configured, the source-based routing and the PBR rules should be configured, similar to the previous section.

Listing 2.27: IRP server IPv4 source-based routing via GRE using the ‘ip‘ command

```
ip route add default dev tun0 table 201
ip route add default dev tun1 table 202
ip route add default dev tun2 table 203
ip rule add from 10.0.1.2 table 201 pref 32101
ip rule add from 10.0.1.6 table 202 pref 32102
ip rule add from 10.0.1.10 table 202 pref 32103
```

Listing 2.28: IRP server IPv4 source-based routing via GRE using standard CentOS configuration files

```
#/etc/sysconfig/network-scripts/route-tun0:
default dev tun0 table 201
default dev tun1 table 202
default dev tun2 table 203
#/etc/sysconfig/network-scripts/rule-tun0:
from 10.0.1.2 table 201 pref 32101
from 10.0.1.6 table 202 pref 32102
from 10.0.1.10 table 203 pref 32103
```

Refer to OS configuration manual for configuration guidelines.

Router configuration looks similar to the previous cases. A Cisco/Brocade example will be provided.

Listing 2.29: Cisco IPv4 PBR configuration: 2 routers, 2 providers, separate Vlan

```
#Router R1
access-list 1 permit ip host 10.0.1.2
access-list 2 permit ip host 10.0.1.6
!
route-map irp-peer permit 10
 match ip address 1
 set ip next-hop 10.11.0.1
 set interface Null0
!
route-map irp-peer permit 20
 match ip address 2
 set ip next-hop 10.12.0.1
 set interface Null0
!

interface Tunnel0
ip policy route-map irp-peer
interface Tunnel1
ip policy route-map irp-peer

#Router R2
access-list 1 permit ip host 10.0.1.10
!
route-map irp-peer permit 10
 match ip address 1
 set ip next-hop 10.13.0.1
 set interface Null0
!
interface Tunnel0
ip policy route-map irp-peer
```

Case 4: Internet Exchanges configuration examples

The PBR rules for Cisco routers/switches are generated by IRP, following the template below.

Listing 2.30: Cisco IPv4 PBR configuration template

```
!--- repeated block for each peering partner
no route-map <ROUTEMAP> permit <ACL>
no ip access-list extended <ROUTEMAP>-<ACL>

ip access-list extended <ROUTEMAP>-<ACL>
  permit ip host <PROBING_IP> any dscp <PROBING_DSCP>

route-map <ROUTEMAP> permit <ACL>
  match ip address <ROUTEMAP>-<ACL>
  set ip next-hop <NEXT_HOP>
  set interface Null0

!--- block at the end of PBR file
interface <INTERFACE>
  ip policy route-map <ROUTEMAP>
```

The “<>” elements represent variables with the following meaning:

- <ROUTEMAP> represents the name assigned by IRP and equals the value of the Route Map parameter in PBR Generator ("irp-ix" in Figure 4)
- <ACL> represents a counter that identifies individual ACL rules. This variable's initial value is taken from ACL name start field of PBR Generator and is subsequently incremented for each ACL
- <PROBING_IP> one of the configured probing IPs that IRP uses to probe link characteristics via different peering partners. One probing IP is sufficient to cover up to 64 peering partners
- <PROBING_DSCP> an incremented DSCP value assigned by IRP for probing a specific peering partner. This is used in combination with the probing IP
- <NEXT_HOP> represents the IP address identifying the peering partner on the exchange. This parameter is retrieved during autoconfiguration and preserved in Exchange configuration
- <INTERFACE> represents the interface where traffic conforming to the rule will exist the Exchange router. This is populated with the Interface value of PBR Generator

The PBR rules for Brocade non-XMR router/switches are generated by IRP, following the template below.

Listing 2.31: Cisco IPv4 PBR configuration template

```
!--- repeated block for each peering partner
no route-map <ROUTEMAP> permit <ACL>
no ip access-list extended <ROUTEMAP>-<ACL>

ip access-list extended <ROUTEMAP>-<ACL>
 permit ip host <PROBING_IP> any dscp-matching <PROBING_DSCP>

route-map <ROUTEMAP> permit <ACL>
 match ip address <ROUTEMAP>-<ACL>
 set ip next-hop <NEXT_HOP>
 set interface Null0

!--- block at the end of PBR file
interface <INTERFACE>
 ip policy route-map <ROUTEMAP>
```

The "<>" elements represent variables with the same meaning as per Cisco example.

Brocade XMR routers use keyword "dscp-mapping" instead of "dscp-matching".

The PBR rules for Juniper routers/switches are generated by IRP, following the template below.

It is important to note that the different sections of the PBR rules (load replace/merge relative terminal) should be entered independently and not as a single file that is output by IRP. Also take note that the last group includes a 'load merge' combo and not a 'load replace' as the first three groups.

Listing 2.32: Juniper IPv4 PBR configuration template

```
load replace relative terminal
[Type ^D at a new line to end input]

interfaces {
 <INTERFACE> {
    unit <INTERFACE_UNIT> {
```

```
            family inet {
                filter {
                    replace:
                    input <ROUTEMAP>;
                }
            }
        }
    }
}


load replace relative terminal
[Type ^D at a new line to end input]

firewall {
    family inet {
        filter <ROUTEMAP> {
            replace:
            term <ROUTEMAP><ACL> {
                from {
                    source-address <PROBING_IP>;
                    dscp <PROBING_DSCP>;
                }
                then {
                    routing-instance <ROUTEMAP><ACL>-route;
                }
            }
...
            replace:
            term default {
                then {
                    accept;
                }
            }
        }
    }
}


load replace relative terminal
[Type ^D at a new line to end input]

routing-instances {
        replace:
        <ROUTEMAP><ACL>-route {
            instance-type forwarding;
            routing-options {
                static {
                    route 0.0.0.0/0 next-hop <NEXT_HOP>;
                }
            }
        }
...
}


load merge relative terminal
[Type ^D at a new line to end input]
```

```
routing-options {
     interface-routes {
        replace:
        rib-group inet <ROUTEMAP>rib;
     }
     rib-groups {
        replace:
        <ROUTEMAP>rib {
           import-rib [ inet.0 <ROUTEMAP><ACL>-route.inet.0 ... ];
        }
     }
}
```

The “<>” elements represent variables with the following meaning:

- <INTERFACE> represents the interface where traffic conforming to the rule will exist the Exchange router. This is populated with the Interface value of PBR Generator
- <INTERFACE_UNIT> is the value of the Interface Unit parameter in PBR Generator
- <ROUTEMAP>represents the name assigned by IRP and equals the value of the Route Map parameter in PBR Generator
- <ACL> represents a combined counter like "00009" that identifies individual ACL rules. This variable's initial value is taken from ACL name start field of PBR Generator and is subsequently incremented for each ACL
- <PROBING_IP> one of the configured probing IPs that IRP uses to probe link characteristics via different peering partners. One probing IP is sufficient to cover up to 64 peering partners
- <PROBING_DSCP> an incremented DSCP value assigned by IRP for probing a specific peering partner. This is used in combination with the probing IP
- <NEXT_HOP> represents the IP address identifying the peering partner on the exchange. This parameter is retrieved during autoconfiguration and preserved in Exchange configuration

Note that Juniper routers/switches need an additional parameter in order to correctly configure PBRs - Interface unit.

Verifying PBR configuration

To ensure that the PBRs are properly configured on the router(s), the standard *NIX traceroute command can be used, by specifying each probing IP as the source IP address for tracing the route. The route should pass through a different provider (note the ISP BGP neighbor IP).

Listing 2.33: PBR validation using ‘traceroute‘

```
root@server ~ $ traceroute -m 5 8.8.8.8 -nns 10.0.0.3
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1 10.0.0.1 0.696 ms  0.716 ms  0.783 ms
 2 10.11.0.1  0.689 ms  0.695 ms  0.714 ms
 3 84.116.132.146 14.384 ms 13.882 ms  13.891 ms
 4 72.14.219.9 13.926 ms 14.477 ms  14.473 ms
 5 209.85.240.64 14.397 ms 13.989 ms  14.462 ms

root@server ~ $ traceroute -m 5 8.8.8.8 -nns 10.0.0.4
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1 10.0.0.1 0.696 ms  0.516 ms  0.723 ms
 2 10.12.0.1  0.619 ms  0.625 ms  0.864 ms
 3 83.16.126.26 13.324 ms 13.812 ms  13.983 ms
```

```
 4 72.14.219.9 15.262 ms 15.347 ms  15.431 ms
 5 209.85.240.64 16.371 ms 16.991 ms  16.162 ms
```

Another useful method for checking that the PBRs are properly configured is to use Explorer self check option. Make sure that the providers are added to IRP configuration before executing Explorer self check.

Listing 2.34: PBR validation using Explorer self check

```
root@server ~ $ /usr/sbin/explorer -s
Starting PBR check

PBR check failed for provider A[2]. Diagnostic hop information: IP
    =10.11.0.12 TTL=3
PBR check succeeded for provider B[3]. Diagnostic hop information: IP
    =10.12.0.1 TTL=3
```

In order to ensure that the PBRs are properly configured for Internet Exchanges, the method below can be used.

Listing 2.35: PBR validation using 'iptables' and 'traceroute'

```
root@server ~ $ iptables -t mangle -I OUTPUT -d 8.8.8.8 -j DSCP --set-dscp
    <PROBING_DSCP>

root@server ~ $ traceroute -m 5 -nns <PROBING_IP> 8.8.8.8
traceroute to 8.8.8.8, 30 hops max, 60 byte packets
 1 ...
 2 ...
 3 <NEXT_HOP> 126.475 ms !X^C
```

where

- <NEXT_HOP> is a Peering Partner's next-hop IP address in IRP configuration
- <PROBING_DSCP> is a Peering Partner's DSCP value in IRP configuration
- <PROBING_IP> is a Peering Partner's probing IP address in IRP configuration

The first IP of the trace leaving client infrastructure should be on the Exchange and the next-hop should belong to the correct Peering Partner.

⚠ iptables rules should be deleted after all tests are done.

2.8.1.1 Current route detection

Current route detection algorithm requires the explorer.infra_ips parameter to be configured. All the IP addresses and subnets belonging to the infrastructure should be added here. These IP addresses can be easily determined by using the `traceroute` command.

2.8.1.2 Providers configuration

Before Explorer starts probing the prefixes detected by the Collector, all providers should be configured.

ℹ For the complete list of provider settings please see the Provider section.
Before configuring providers in IRP, the BGP sessions need to be defined, see Bgpd Configuration

⛔ Please note that some Brocade router models do not support SNMP counters per Vlan, therefore physical interfaces should be used for gathering traffic statistics.

For the sample configuration below, let's assume the following:

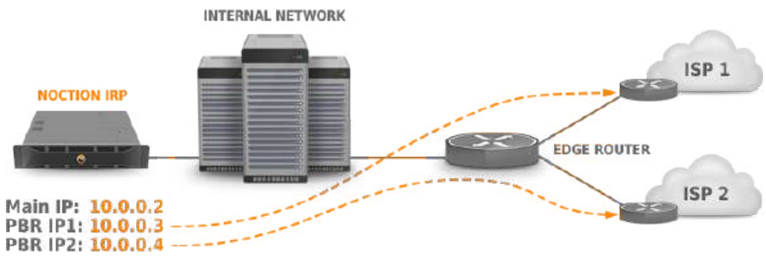

Figure 2.8.4: PBR configuration: single router and separate probing Vlan

`ISP1` - the provider's name

`10.0.0.1` - Router IP configured on the probing Vlan

`10.0.0.3` - Probing IP for ISP1, configured on the IRP server

`10.11.0.1, 10.11.0.2` - IP addresses used for the EBGP session with the ISP, 10.11.0.2 being configured on the router

`400Mbps` - the agreed bandwidth

`1Gbps` - the physical interface throughput

`'public'` - read-only SNMP community configured on R1

`GigabitEthernet2/1` - the physical interface that connects R1 to ISP1

As presented in Listing 2.36, all the parameters are pretty self-explanatory. Please check the BGP Monitoring section as well.

Listing 2.36: Sample provider configuration

```
peer.1.95th = 400
peer.1.95th.bill_day = 1
peer.1.bgp_peer = R1
peer.1.cost = 6
peer.1.description = ISP1
peer.1.ipv4.next_hop = 10.11.0.1
peer.1.ipv4.probing = 10.0.0.3
peer.1.ipv4.diag_hop = 10.11.0.1
peer.1.ipv4.mon = 10.11.0.1 10.11.0.2
peer.1.limit_load = 1000
peer.1.shortname = ISP1
peer.1.snmp.interfaces = 1:GigabitEthernet2/1
peer.1.mon.ipv4.bgp_peer  = 10.11.0.1

snmp.1.name = Host1
snmp.1.ip = 10.0.0.1
snmp.1.community = public
```

SNMP parameters validation

To make sure that the SNMP parameters are correct, the 'snmpwalk' tool can be run on the IRP server:

Listing 2.37: SNMP parameters validation

```
root@server ~ $ snmpwalk -v2c -c irp-public 10.0.0.1 ifDescr
IF-MIB::ifDescr.1 = STRING: GigabitEthernet1/1
IF-MIB::ifDescr.2 = STRING: GigabitEthernet1/2
IF-MIB::ifDescr.3 = STRING: GigabitEthernet2/1
IF-MIB::ifDescr.4 = STRING: GigabitEthernet2/2
IF-MIB::ifDescr.5 = STRING: GigabitEthernet2/3
IF-MIB::ifDescr.6 = STRING: GigabitEthernet2/4

root@server ~ $ snmpwalk -v2c -c irp-public 10.0.0.1 ifIndex
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifIndex.3 = INTEGER: 3
IF-MIB::ifIndex.4 = INTEGER: 4
IF-MIB::ifIndex.5 = INTEGER: 5
IF-MIB::ifIndex.6 = INTEGER: 6
```

2.8.2 Flowspec PBR

In networks that have Flowspec and Redirect to IP capabilities, PBR can be implemented by means of Flowspec policies. Refer details in Flowspec policies.

In order to use assigned providers or peers on Internet Exchanges in IRP the same set of source IP addresses and DSCP values (for Internet Exchanges) are assigned to individual providers/peers. These values are used to automatically generate Flowspec policies that redirect IRP probes to designated next-hops.

In order to use this feature global.flowspec, global.flowspec.pbr and bgpd.peer.X.flowspecmust be enabled.

⚠ Flowspec PBR cannot be used during non-intrusive (global.nonintrusive_bgp) mode.

2.9 Bgpd Configuration

For IRP to inject the improved prefixes to the routing tables, proper iBGP sessions have to be configured between the edge routers and the IRP. The IRP BGP daemon acts similarly to a *route-reflector-client*. Following criteria need to be met:

1. An internal BGP session using the same autonomous system number (ASN) must be configured between each edge router and the IRP. BGP sessions must not be configured with *next-hop-self* (route reflectors can't be used to inject routes with modified *next-hop*) - the *next-hop* parameter advertised by IRP Bgpd should be distributed to other iBGP neighbors.

2. *route-reflector-client* must be enabled for the routes advertised by IRP Bgpd to be distributed to all non-client neighbors.

3. Routes advertised by IRP Bgpd must have a higher preference over routes received from external BGP neighbors.
   This can be done by different means, on the IRP or on the router side:

   - *Local-pref* can be set to a reasonably high value in the Bgpd configuration
   - *Communities* can be appended to prefixes advertised by Bgpd

   Avoid colisions of localpref or communities values assigned to IRP within both its configuration and/or on customer's network.

   - Multi-exit-discriminator (*MED*) can be changed to affect the best-path selection algorithm
   - *Origin* of the advertised route can be left unchanged or overridden to a specific value (*incomplete, IGP, EGP*)

   LocalPref, MED and Origin attribute values are set with the first nonempty value in this order: 1) value from configuration or 2) value taken from incoming aggregate or 3) default value specified in RFC4271.

   Communities attribute value concatenates the value taken from incoming aggregate with configuration value. The router should be configured to send no Communities attribute in case it is required that IRP announces Communities attribute that contain only the configured value.

4. BGP *next-hop* must be configured for each provider configured in IRP (please refer to Providers configuration and Provider)

Special attention is required if edge routers are configured to announce prefixes with empty AS-Path. In some cases, improvements announced by Bgpd may have an empty AS-Path. Thus, when edge router does not have any prefix-list filtering enforced, all the current improvements can be improperly advertised to routers - this may lead to policy violations and session reset. Refer to AS-Path behavior in IRP Bgpd regarding the way to disallow announcements with empty AS-Path.

None of the improvements advertised by IRP should be advertised to your external peers (refer to bgpd.no_export).

We recommend the routes to be injected into the edge router which runs the BGP session with the provider. This ensures that the routes are properly redistributed across the network.

For example, there are two routers: R1 and R2. R1 runs a BGP session with Level3 and R2 with Cogent. The current route is x.x.x.x/24 with the next-hop set to Level3 and all the routers learn this route via R1. The system injects the x.x.x.x/24 route to R2 with the next-hop updated to Cogent.

In this case the new route is installed on the routing table and it will be properly propagated to R1 (and other routers) via iBGP.

However if the system injects the new route to R1 instead of R2, the route's next-hop will point to R2 while R2 will have the next-hop pointing to R1 as long as the injected route is propagated over iBGP to other routers. In this case a routing loop will occur.

In the following BGP session configuration example, the IRP server with IP `10.0.0.2` establishes an iBGP session to the edge router (IP: `10.0.0.1`). The *local-pref* parameter for the prefixes advertised by IRP is set to 190. BGP monitoring (see BGP Monitoring, Bgpd settings) is enabled.

Listing 2.38: iBGP session configuration example

```
bgpd.peer.R1.as              = 65501
bgpd.peer.R1.our_ip          = 10.0.0.2
bgpd.peer.R1.master_peer_ip        = 10.0.0.1
bgpd.peer.R1.listen  = 1
bgpd.peer.R1.localpref       = 190
bgpd.peer.R1.shutdown = 0
```

Vendor-specific router-side iBGP session configuration examples:

Vyatta routers:

Listing 2.39: Vyatta IPv4 iBGP session configuration example

```
set protocols bgp 65501 neighbor 10.0.0.2 remote-as '65501'
set protocols bgp 65501 neighbor 10.0.0.2 route-reflector-client
set protocols bgp 65501 parameters router-id '10.0.0.1'
```

Listing 2.40: Vyatta IPv6 iBGP session configuration example

```
delete system ipv6 disable-forwarding
commit
set protocols bgp 65501 neighbor 2001:db8:2::2 remote-as '65501'
set protocols bgp 65501 neighbor 2001:db8:2::2 route-reflector-client
set protocols bgp 65501 neighbor 2001:db8:2::2 address-family 'ipv6-unicast
    '
set protocols bgp 65501 parameters router-id '10.0.0.1'
```

Listing 2.41: Vyatta IPv4 route-map for setting local-pref on the router

```
set protocols bgp 65501 neighbor 10.0.0.2 route-map import 'RM-IRP-IN'
set policy route-map RM-IRP-IN rule 10 action 'permit'
set policy route-map RM-IRP-IN rule 10 set local-preference '190'
```

Listing 2.42: Vyatta IPv6 route-map for setting local-pref on the router

```
set protocols bgp 65501 neighbor 2001:db8:2::2 route-map import 'RM-IRP-IN'
set policy route-map RM-IRP-IN rule 10 action 'permit'
set policy route-map RM-IRP-IN rule 10 set local-preference '190'
```

Cisco routers:

Listing 2.43: Cisco IPv4 iBGP session configuration example

```
router bgp 65501
neighbor 10.0.0.2 remote-as 65501
neighbor 10.0.0.2 send-community
neighbor 10.0.0.2 route-reflector-client
```

Listing 2.44: Cisco IPv6 iBGP session configuration example

```
router bgp 65501
neighbor 2001:db8:2::2 remote-as 65501
neighbor 2001:db8:2::2 send-community
neighbor 2001:db8:2::2 route-reflector-client

or

router bgp 65501
neighbor 2001:db8:2::2 remote-as 65501
no neighbor 2001:db8:2::2 activate
address-family ipv6
neighbor 2001:db8:2::2 activate
neighbor 2001:db8:2::2 send-community
neighbor 2001:db8:2::2 route-reflector-client
```

Listing 2.45: Cisco IPv4 route-map for setting local-pref on the router

```
router bgp 65501
neighbor 10.0.0.2 route-map RM-IRP-IN input
route-map RM-IRP-IN permit 10
 set local-preference 190
```

Listing 2.46: Cisco IPv6 route-map for setting local-pref on the router

```
router bgp 65501
neighbor 2001:db8:2::2 route-map RM-IRP-IN input
route-map RM-IRP-IN permit 10
 set local-preference 190
```

Listing 2.47: Limiting the number of received prefixes for an IPv4 neighbor on Cisco

```
router bgp 65501
neighbor 10.0.0.2 maximum-prefix 10000
```

Listing 2.48: Limiting the number of received prefixes for an IPv6 neighbor on Cisco

```
router bgp 65501
neighbor 2001:db8:2::2 maximum-prefix 10000
```

Juniper equipment:

The Cluster ID must be unique in multi-router configuration. Otherwise improvements will not be redistributed properly. Cluster ID is optional for single router networks.

Listing 2.49: Juniper IPv4 iBGP session configuration example

```
[edit]
routing-options {
    autonomous-system 65501;
    router-id 10.0.0.1;
}
protocols {
    bgp {
        group 65501 {
            type internal;
         cluster 0.0.0.1;
```

```
         family inet {
                unicast;
            }
            peer-as 65501;
            neighbor 10.0.0.2;
        }
    }
}
```

Listing 2.50: Juniper IPv6 iBGP session configuration example

```
[edit]
routing-options {
    autonomous-system 65501;
    router-id 10.0.0.1;
}
protocols {
    bgp {
        group 65501 {
            type internal;
         cluster 0.0.0.1;
            family inet6 {
                any;
            }
            peer-as 65501;
            neighbor 2001:db8:2::2;
        }
    }
}
```

Listing 2.51: Juniper IPv4 route-map for setting local-pref on the router

```
[edit]
routing-options {
    autonomous-system 65501;
    router-id 10.0.0.1;
}
protocols {
    bgp {
        group 65501 {
            type internal;
            peer-as 65501;
            neighbor 10.0.0.2 {
                preference 190;
            }
        }
    }
}
```

Listing 2.52: Limiting the number of received prefixes for an IPv4 neighbor on Juniper

```
protocols {
    bgp {
        group 65501 {
            neighbor 10.0.0.2 {
                family inet {
                    any {
                        prefix-limit {
                            maximum 10000;
```

```
                                        teardown;
                                    }
                                }
                            }
                        }
                    }
                }
}
```

2.9.1 AS-Path behavior in IRP Bgpd

Every advertised prefix contains an AS-Path attribute. However, in some cases this attribute can be empty.

For compatibility purposes, the IRP Bgpd has a few algorithms handling the AS-Path attribute:

1. The advertised prefix will be marked with a recovered AS-Path attribute.
   Recovered AS-Path is composed of consecutive AS-Numbers that are collected during exploring process. Please note that recovered AS-Path may differ from the actual BGP Path.

2. The advertised prefix will be marked with the AS-Path from the aggregate received via BGP.

3. If the advertised prefix, for whatever reason, has an empty AS-Path, it can be announced or ignored, depending on the Bgpd configuration.

For a detailed description refer to bgpd.as_path.

In case certain routes should be preferred over decisions made by IRP, use one of the following ways:

- Routers may be configured to have more preferable localpref / weight values for such routes so the best path algorithm always selects these routes instead of the routes injected by the Bgpd daemon.

- Routes may be filtered or have lower localpref / weight set up, using incoming route-map applied to BGP session with IRP.

- Networks that must be completely ignored by IRP can be specified in global.ignored.asn, global.ignorednets parameters or marked with a BGP Community listed in global.ignored_communities, so no probing / improving / announcing will be made by IRP.

If split announcements are enabled in IRP (refer 4.4.37) the routes announced by IRP will be more specific.

2.9.2 Bgpd online reconfiguration

Bgpd can be reconfigured without BGP sessions restart. This feature increases network stability and reduces routers' CPU load during session initiation.

Use the commands below to inform IRP Bgpd to reload its configuration:

Listing 2.53: Reload IRP Bgpd configuration

```
root@server ~ $ systemctl reload bgpd
```

2.9.3 BGP Additional paths

Typically router sends only the best routes and that behaviour hides alternatives.

Enabling sending of additional paths (RFC 7911) allows IRP to automatically configure complete list of routes for partial routing providers and internet exchanges peering partners.

Listing 2.54: Cisco: Configure BGP Additional Paths

```
router bgp 65500
neighbor 10.1.1.2 remote-as 65500
!
address-family ipv4
 bgp additional-paths select all
 neighbor 10.1.1.2 activate
 neighbor 10.1.1.2 additional-paths send
 neighbor 10.1.1.2 advertise additional-paths all
exit-address-family
```

Listing 2.55: Juniper: Configure BGP Additional Paths

```
[edit protocols bgp group group-name family family]
add-path {
  send {
    path-count 20;
  }
}
```

2.10 BMP Configuration

IRP's BMP monitoring station passively listens on a designated TCP port for monitoring routers to establish BMP sessions. Further BMP setup is performed on monitoring router where

- BMP monitoring station's IP address and port are set and also
- filtering rules regarding what BMP data is sent to IRP are applied if needed.

By default BMP implementations do not filter routing data sent to a monitoring station (refer to peer.X.bmp.check_routes).

BMP related features are configured individually for example:

- BMP monitoring station: BMP monitoring station settings.
- primary source of data for current route re-construction: bgpd.as_path.

It is recommended that BMP is set as the primary source for current route reconstruction only when BMP data is available for all providers.

- improvement old and new provider re-probing on AS Path changes: bgpd.retry_probing.new.bmp_path_change, bgpd.retry_probing.old.bmp_path_change.

2.11 Threat Mitigation Configuration

2.11.1 BGP Blackholing

BGP Blackholing requires the following parameters to be configured:

- Blackholing next-hop bgpd.peer.X.blackholing.ipv4.next_hop & bgpd.peer.X.blackholing.ipv6.next_hop
- Each eligible provider should have a configured Blackholing BGP session peer.X.blackholing.bgp_peer and Blackholing community peer.X.blackholing.community (formerly offered by a provider)
- Default BGP reaction (irpdetectd.bgp.reaction) should be set to 0 (Drop)
- Moderated/Automated Mode (irpdetectd.mode) requires thresholds to be set to operate properly (irpdetectd.blackhole.threshold.kpps and/or irpdetectd.blackhole.threshold.mbps).

2.11.2 BGP Redirect

BGP Redirect requires the following parameters to be configured:

- Blackholing next-hop bgpd.peer.X.blackholing.ipv4.next_hop & bgpd.peer.X.blackholing.ipv6.next_hop
- BGP redirect routers (irpdetectd.bgp.redirect.bgp_peers) and BGP redirect communities (irpdetectd.bgp.redirect.communities)
- Default BGP reaction (irpdetectd.bgp.reaction) should be set to 1 (Redirect)
- Moderated/Automated Mode (irpdetectd.mode) requires thresholds to be set to operate properly (irpdetectd.blackhole.threshold.kpps and/or irpdetectd.blackhole.threshold.mbps).

2.11.3 FlowSpec Drop

FlowSpec Drop requires the following parameters to be configured:

- Configure FlowSpec globally and enable on each required BGP session (bgpd.peer.X.flowspec)
- Default FlowSpec reaction (irpdetectd.flowspec.reaction) should be set to 0 (Drop)
- Moderated/Automated Mode (irpdetectd.mode) requires thresholds to be set to operate properly (irpdetectd.flowspec.threshold.kpps and/or irpdetectd.flowspec.threshold.mbps).

2.11.4 FlowSpec Redirect

FlowSpec Redirect requires the following parameters to be configured:

- Configure FlowSpec globally and enable on each required BGP session (bgpd.peer.X.flowspec)
- FlowSpec redirect IPv4 (irpdetectd.flowspec.ipv4.redirect)
- FlowSpec redirect IPv6 (irpdetectd.flowspec.ipv6.redirect)
- Default FlowSpec reaction (irpdetectd.flowspec.reaction) should be set to 1 (Redirect)
- Moderated/Automated Mode (irpdetectd.mode) requires thresholds to be set to operate properly (irpdetectd.flowspec.threshold.kpps and/or irpdetectd.flowspec.threshold.mbps).

2.11.5 Mitigation prefix size (IPv4 / IPv6)

Threat Mitigation can be configured to perform an action against the attacked host or subnet.

The size of the subnet mask for the announced IPv4 prefixes (irpdetectd.ipv4.prefix_size) and IPv6 prefixes (irpdetectd.ipv6.prefix_size) can be adjusted to the desired value.

2.12 Administrative Components

There are two additional components in the IRP system: **dbcron** and **irpmng**. For detailed configuration parameters description, please see the Administrative settings section.

Dbcron handles periodic database maintenance tasks and statistics processing.

Irpmng performs various management tasks and provides a CLI interface to query certain IRP data.

2.13 Failover Configuration

Failover relies on many operating system, networking and database components to work together. All these must be planned in advance and require careful implementation.

We recommend that you request Noction's engineers to assist with failover configuration.

Subsequent sections should be considered in presented order when configuring a fresh IRP failover setup. Refer to them when troubleshooting or restoring services.

2.13.1 Initial failover configuration

Prerequisites

Before proceeding with failover configuration the following prerequisites must be met:

- One IRP node is configured and fully functional. We will refer to this node as $IRPMASTER.
- Second IRP node is installed with the same version of IRP as on $IRPMASTER. We will refer to this node as $IRPSLAVE.

Second IRP node MUST run the same operating system as $IRPMASTER.

When troubleshooting problems, besides checking matching IRP versions ensure the same versions of irp MySQL databases are installed on both failover nodes.

- IRP services, MySQL and HTTP daemons are stopped on $IRPSLAVE node.
- Network operator can SSH to both $IRPMASTER and $IRPSLAVE and subsequent commands are assumed to be run from a $IRPMASTER console.

$IRPMASTER and $IRPSLAVE must have different hostnames.

Configure communication channel from $IRPMASTER to $IRPSLAVE

This channel is used during failover configuration and subsequently $IRPMASTER uses it to synchronize IRP configuration changes when these occur. It uses key-based authentication without a passphrase to allow automated logins on $IRPSLAVE by $IRPMASTER processes.

Adjust firewalls if any so that $IRPMASTER node can access $IRPSLAVE via SSH.

Generate SSH keys pair WITHOUT passphrase:

Listing 2.56: Generate keys on $IRPMASTER

```
root@IRPMASTER ~ # ssh-keygen -t rsa -b 2048 -f ~/.ssh/id_rsa -C "
    failover@noction"
```

Default keys files are used. In case your system needs additional keys for other purposes we advise that those keys are assigned a different name. If this is not possible then keys file name designated for failover use should be also specified in IRP configuration parameter global.failover_identity_file.

Copy public SSH key from master to slave instance:

Listing 2.57: Install public key on $IRPSLAVE

```
root@IRPMASTER ~ # cat ~/.ssh/id_rsa.pub | while read key; do ssh $IRPSLAVE
     "echo $key >> ~/.ssh/authorized_keys"; done
```

Check if $IRPMASTER can login to $IRPSLAVE without using a password:

Listing 2.58: Check SSH certificate-based authentication works

```
root@IRPMASTER ~ # ssh $IRPSLAVE
```

Install certificate and keys for MySQL Multi-Master replication between $IRPMASTER and $IRPSLAVE

MySQL Multi-Master replication uses separate communication channels that also require authentication. IRP failover uses key-based authentication for these channels too.

Adjust firewalls if any so that $IRPMASTER and $IRPSLAVE can communicate with each other bidirectionally.

Create Certificate Authority and server/client certificates and keys. Commands must be run on both $IRPMASTER and $IRPSLAVE nodes:

Listing 2.59: Generate CA and certificates

```
# cd && rm -rvf irp-certs && mkdir -p irp-certs && cd irp-certs

# openssl genrsa 2048 > $(hostname -s)-ca-key.pem

# openssl req -new -x509 -nodes -days 3600 -subj "/C=US/ST=CA/L=Palo Alto/O
    =Noction/OU=Intelligent Routing Platform/CN=$(/bin/hostname) CA/
    emailAddress=support@noction.com" -key $(hostname -s)-ca-key.pem -out $(
    hostname -s)-ca-cert.pem

# openssl req -newkey rsa:2048 -days 3600 -subj "/C=US/ST=CA/L=Palo Alto/O=
    Noction/OU=Intelligent Routing Platform/CN=$(/bin/hostname) server/
    emailAddress=support@noction.com" -nodes -keyout $(hostname -s)-server-
    key.pem -out $(hostname -s)-server-req.pem

# openssl rsa -in $(hostname -s)-server-key.pem -out $(hostname -s)-server-
    key.pem
```

```
# openssl x509 -req -in $(hostname -s)-server-req.pem -days 3600 -CA $(
    hostname -s)-ca-cert.pem -CAkey $(hostname -s)-ca-key.pem -set_serial 01
     -out $(hostname -s)-server-cert.pem

# openssl req -newkey rsa:2048 -days 3600 -subj "/C=US/ST=CA/L=Palo Alto/O=
    Noction/OU=Intelligent Routing Platform/CN=$(/bin/hostname) client/
    emailAddress=support@noction.com" -nodes -keyout $(hostname -s)-client-
    key.pem -out $(hostname -s)-client-req.pem

# openssl rsa -in $(hostname -s)-client-key.pem -out $(hostname -s)-client-
    key.pem

# openssl x509 -req -in $(hostname -s)-client-req.pem -days 3600 -CA $(
    hostname -s)-ca-cert.pem -CAkey $(hostname -s)-ca-key.pem -set_serial 01
     -out $(hostname -s)-client-cert.pem
```

Verify certificates. Commands must be run on both $IRPMASTER and $IRPSLAVE nodes:

Listing 2.60: Verify certificates

```
# openssl verify -CAfile $(hostname -s)-ca-cert.pem $(hostname -s)-server-
    cert.pem $(hostname -s)-client-cert.pem

server-cert.pem: OK
client-cert.pem: OK
```

Install certificates in designated directories. Commands must be run on both $IRPMASTER and $IRP-SLAVE nodes:

Listing 2.61: Install certificates in designated directories

```
# mkdir -p /etc/pki/tls/certs/mysql/server/ /etc/pki/tls/certs/mysql/client
    / /etc/pki/tls/private/mysql/server/ /etc/pki/tls/private/mysql/client/

# cp $(hostname -s)-ca-cert.pem $(hostname -s)-server-cert.pem /etc/pki/tls
    /certs/mysql/server/
# cp $(hostname -s)-ca-key.pem $(hostname -s)-server-key.pem /etc/pki/tls/
    private/mysql/server/
# cp $(hostname -s)-client-cert.pem /etc/pki/tls/certs/mysql/client/
# cp $(hostname -s)-client-key.pem /etc/pki/tls/private/mysql/client/

# cd && rm -rvf irp-certs
```

Cross copy client key and certificates:

Listing 2.62: Cross copy client key and certificates

```
root@IRPMASTER ~# scp "/etc/pki/tls/certs/mysql/server/$IRPMASTER-ca-cert.
    pem" "$IRPSLAVE:/etc/pki/tls/certs/mysql/client/"
root@IRPMASTER ~# scp "/etc/pki/tls/certs/mysql/client/$IRPMASTER-client-
    cert.pem" "$IRPSLAVE:/etc/pki/tls/certs/mysql/client/"
root@IRPMASTER ~# scp "/etc/pki/tls/private/mysql/client/$IRPMASTER-client-
    key.pem" "$IRPSLAVE:/etc/pki/tls/private/mysql/client/"

root@IRPMASTER ~# scp "$IRPSLAVE:/etc/pki/tls/certs/mysql/server/$IRPSLAVE-
    ca-cert.pem" "/etc/pki/tls/certs/mysql/client/"
root@IRPMASTER ~# scp "$IRPSLAVE:/etc/pki/tls/certs/mysql/client/$IRPSLAVE-
    client-cert.pem" "/etc/pki/tls/certs/mysql/client/"
```

```
root@IRPMASTER ~# scp "$IRPSLAVE:/etc/pki/tls/private/mysql/client/
    $IRPSLAVE-client-key.pem" "/etc/pki/tls/private/mysql/client/"
```

Adjust file permissions. Commands must be run on both $IRPMASTER and $IRPSLAVE nodes:

Listing 2.63: Set file permissions for keys and certificates

```
# chown -R mysql:mysql /etc/pki/tls/certs/mysql/ /etc/pki/tls/private/mysql
    /
# chmod 0600 /etc/pki/tls/private/mysql/server/* /etc/pki/tls/private/mysql
    /client/*
```

Configure MySQL replication on $IRPSLAVE

Each node of MySQL Multi-Master replication is assigned its own identifier and references previously configured certificates. There are also configuration parameters such as binary/relay log file names and auto-increment and auto-increment-offset values.

IRP includes a template config file /usr/share/doc/irp/irp.my_repl_slave.cnf.template. The template designates $IRPSLAVE as second server of the Multi-Master replication and includes references to $(hostname -s) that need to be replaced with the actual hostname of $IRPSLAVE before installing. Apply the changes and review the configuration file. Alternatively a command like in the below example can be used to create $IRPSLAVE config file from template. Ensure using actual short host name instead of the provided variable:

Listing 2.64: Example $IRPSLAVE configuration from template

```
# Ubuntu
root@IRPSLAVE ~# sed 's|$(hostname -s)|$IRPSLAVE|' < /usr/share/doc/irp/irp
    .my_repl_slave.cnf.template > /etc/mysql/conf.d/irp.my_repl_slave.cnf
# RedHat
root@IRPSLAVE ~# sed 's|$(hostname -s)|$IRPSLAVE|' < /usr/share/doc/irp/irp
    .my_repl_slave.cnf.template > /etc/my.cnf.d/irp.my_repl_slave.cnf
```

The config file created above must be included into $IRPSLAVE node's MySQL config my.cnf. It is recommended to store these files inside OS-specific directories for MariaDB configuration files (Ubuntu: `/etc/mysql/conf.d`, RedHat: `/etc/my.cnf.d/`, otherwise it should be included via `!include /path/to/file` from main MariaDB config.

Check Multi-Master configuration on $IRPSLAVE:

Listing 2.65: Check MySQL on $IRPSLAVE works correctly

```
root@IRPSLAVE ~# systemctl start mariadb
root@IRPSLAVE ~# tail -f /var/log/mysqld.log
root@IRPSLAVE ~# mysql irp -e "show master status \G"
root@IRPSLAVE ~# systemctl stop mariadb
```

Configure MySQL replication on $IRPMASTER

Configuring $IRPMASTER is done with running services.

⚠ Configuring MySQL Multi-Master replication on $IRPMASTER should only be done after confirming it works on $IRPSLAVE.

Similarly to $IRPSLAVE above IRP comes with a template configuration file for $IRPMASTER - /usr/share/doc/irp/irp.my_repl_master.cnf.template. The template designates $IRPMASTER as first server of the Multi-Master replication and includes references to $(hostname -s) that need to be replaced with the actual hostname of $IRPMASTER before installing. Apply the changes and review the resulting configuration file.

Alternatively a command like the example below can be used to create $IRPMASTER config file from template. Ensure using actual short host name instead of the provided variable

Listing 2.66: Set $IRPMASTER as a first node for Multi-Master replication

```
# Ubuntu
root@IRPMASTER ~# sed 's|$(hostname -s)|$IRPMASTER|' < /usr/share/doc/irp/
    irp.my_repl_master.cnf.template > /etc/mysql/conf.d/irp.my_repl_master.
    cnf
# RedHat
root@IRPMASTER ~# sed 's|$(hostname -s)|$IRPMASTER|' < /usr/share/doc/irp/
    irp.my_repl_master.cnf.template > /etc/my.cnf.d/irp.my_repl_master.cnf
```

Again, the config file created above must be included into $IRPSLAVE node's MySQL config my.cnf. It is recommended to store these files inside OS-specific directories for MariaDB configuration files (Ubuntu: `/etc/mysql/conf.d`, RedHat: `/etc/my.cnf.d/`, otherwise it should be included via `!include /path/to/file` from main MariaDB config.

Check MySQL Multi-Master configuration on $IRPMASTER:

Listing 2.67: Check MySQL on $IRPMASTER works correctly

```
root@IRPMASTER ~# systemctl restart mariadb
root@IRPMASTER ~# tail -f /var/log/mysqld.log
root@IRPMASTER ~# mysql irp -e "show master status \G"
```

⛔ If Multi-Master configuration on $IRPMASTER fails or causes unrecoverable errors, a first troubleshooting step is to comment back the included line in /etc/my.cnf on master and slave and restart mysqld service to revert to previous good configuration.

Create replication grants on $IRPMASTER

Replication requires a MySQL user with corresponding access rights to replicated databases. The user must be assigned a password and is designated as connecting correspondingly from $IRPMASTER or $IRPSLAVE. Unfortunately hostnames cannot be used for this and the exact IP addresses of the corresponding nodes must be specified.

⚠ The user is created only once in our procedure since after being created the database on $IRPMASTER is manually transferred to $IRPSLAVE and the user will be copied as part of this process.

Grant replication access to replication user:

Listing 2.68: Replication user and grants

```
mysql> CREATE USER 'irprepl'@'<mysql_slave1_ip_address>' IDENTIFIED BY '<
    replication_user_password>';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'irprepl'@'<
    mysql_masterslave1_ip_address>' REQUIRE CIPHER 'DHE-RSA-AES256-SHA';
mysql> CREATE USER 'irprepl'@'<mysql_master2_ip_address>' IDENTIFIED BY '<
    replication_user_password>';
```

```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'irprepl'@'<
    mysql_slave2_ip_address>' REQUIRE CIPHER 'DHE-RSA-AES256-SHA';
```

Copy IRP database configuration and database to $IRPSLAVE

IRP failover copies the irp database and corresponding configuration file from $IRPMASTER to $IRPSLAVE before activating second server. This avoids the need to manually verify and synchronize identifiers.

Copy root's .my.cnf config file if exists:

Listing 2.69: Copy database root user configuration file

```
root@IRPMASTER ~# scp /root/.my.cnf $IRPSLAVE:/root/
```

Copy config files:

Listing 2.70: Copy database configuration files

```
root@IRPMASTER ~# scp /etc/noction/db.global.conf $IRPSLAVE:/etc/noction/
root@IRPMASTER ~# scp /etc/noction/clickhouse/users.xml $IRPSLAVE:/etc/
    noction/clickhouse/
```

Preliminary copy database files:

Listing 2.71: Copy database data files

```
root@IRPMASTER ~# rsync -av --progress --delete --delete-after --exclude="
    master.info" --exclude="relay-log.info" --exclude="*-bin.*" --exclude
    ="*-relay.*" /var/lib/mysql/ $IRPSLAVE:/var/lib/mysql/
```

Preliminary copy ensures that large files that take a long time to copy are synced to $IRPSLAVE without stopping MySQL daemon on $IRPMASTER and only a reduced number of differences will need to by synced while MySQL is stopped. This operation can be rerun one more time to reduce the duration of the downtime on $IRPMASTER even more.

Finish copy of database files (OS with Systemd):

Listing 2.72: Copy differences of database files (OS with Systemd)

```
root@IRPMASTER ~# systemctl stop mariadb clickhouse-server # RedHat
    Enterprise Linux
root@IRPMASTER ~# systemctl stop mysql clickhouse-server # Ubuntu
root@IRPMASTER ~# systemctl start irp-stop-nobgpd.target

systemctl start irp-shutdown-except-bgpd.target
systemctl start irp-shutdown.target

root@IRPMASTER ~# cd /var/lib/mysql && rm -vf ./master.info ./relay-log.
    info ./*-bin.* ./*-relay.*
root@IRPMASTER ~# rsync -av --progress --delete --delete-after /var/lib/
    mysql/ $IRPSLAVE:/var/lib/mysql/
```

The procedure above tries to reduce the downtime of MySQL daemon on $IRPMASTER. During this time Bgpd preserves IRP Improvements. Make sure this action takes less than bgpd.db.timeout.withdraw.

Start MySQL on $IRPSLAVE and check if there are no errors at /var/log/mysqld.log on RedHat Enterprise Linux or /var/log/mysql/error.log on Ubuntu.

First $IRPSLAVE must be checked.

Start MySQL on IRP master and check if there are no errors in MySQL logs as above.

Start replication (Slaves) on both $IRPMASTER and $IRPSLAVE

MySQL Multi-Master replication uses a replication scheme where a replicating master acts as both replication master and replication slave. The steps above configured IRP nodes to be capable of acting as replication masters. Here we ensure that both nodes are capable of acting as replication slaves.

IRP provides a template command that needs to be adjusted for each $IRPMASTER and $IRPSLAVE and will instruct MySQL daemon to take the replication slave role. The template is /usr/share/doc/irp/changemasterto.template.

The template generates a different command for each $IRPMASTER and $IRPSLAVE nodes and requires multiple values to be reused from configuration settings described above. The command that is run on one node points to the other node as its master.

Make $IRPMASTER a replication slave for $IRPSLAVE:

Listing 2.73: Set $IRPMASTER as replication slave

```
$IRPMASTER-mysql>
CHANGE MASTER TO
MASTER_HOST='$IRPSLAVE-ip-address',
MASTER_USER='irprepl',
MASTER_PASSWORD='$IRPSLAVE-password>',
MASTER_PORT=3306,
MASTER_LOG_FILE= '$IRPSLAVE--bin.000001',
MASTER_LOG_POS= <$IRPSLAVE-bin-log-position>,
MASTER_CONNECT_RETRY=10,
MASTER_SSL=1,
MASTER_SSL_CAPATH='/etc/pki/tls/certs/mysql/client/',
MASTER_SSL_CA='/etc/pki/tls/certs/mysql/client/$IRPSLAVE-ca-cert.pem',
MASTER_SSL_CERT='/etc/pki/tls/certs/mysql/client/$IRPSLAVE-client-cert.pem'
    ,
MASTER_SSL_KEY='/etc/pki/tls/private/mysql/client/$IRPSLAVE-client-key.pem'
    ,
MASTER_SSL_CIPHER='DHE-RSA-AES256-SHA';
```

You must manually check what values to assign to
$IRPSLAVE-bin.000001 and <$IRPSLAVE-bin-log-position>
by running the following MySQL command on $IRPSLAVE
mysql> show master status

For the initial configuration the values for $IRPSLAVE-bin.000001 and <$IRPSLAVE-bin-log-position> must be as follows:

Binlog file: $IRPSLAVE–bin.000001

Binlog position: 106

Run the commands below to run the replication and check the slave status:

Listing 2.74: Starting replication slave on $IRPMASTER

```
mysql> START SLAVE \G
mysql> show slave status \G
```

Check the Slave_IO_State, Last_IO_Errno, Last_IO_Error, Last_SQL_Errno, Last_SQL_Error values for errors. Make sure there are no errors.

Make $IRPSLAVE a replication slave for $IRPMASTER:

Listing 2.75: Set $IRPSLAVE as replication slave

```
$IRPSLAVE-mysql>
CHANGE MASTER TO
MASTER_HOST='$IRPMASTER-ip-address',
MASTER_USER='irprepl',
MASTER_PASSWORD='$IRPMASTER-password>',
MASTER_PORT=3306,
MASTER_LOG_FILE= '$IRPMASTER-bin.000001',
MASTER_LOG_POS= <$IRPMASTER-bin-log-position>,
MASTER_CONNECT_RETRY=10,
MASTER_SSL=1,
MASTER_SSL_CAPATH='/etc/pki/tls/certs/mysql/client/',
MASTER_SSL_CA='/etc/pki/tls/certs/mysql/client/$IRPMASTER-ca-cert.pem',
MASTER_SSL_CERT='/etc/pki/tls/certs/mysql/client/$IRPMASTER-client-cert.pem
    ',
MASTER_SSL_KEY='/etc/pki/tls/private/mysql/client/$IRPMASTER-client-key.pem
    ',
MASTER_SSL_CIPHER='DHE-RSA-AES256-SHA';
```

You must manually check what values to assign to

$IRPMASTER-bin.000001and <$IRPMASTER-bin-log-position>

by running the following MySQL command on $IRPMASTER

mysql> show master status

For the initial configuration the values for $IRPMASTER-bin.000001 and <$IRPMASTER-bin-log-position> must be as follows:

Binlog file: $IRPMASTER–bin.000001

Binlog position: 106

Run the commands below to run the replication and check the slave status:

Listing 2.76: Starting replication slave

```
mysql> START SLAVE \G
mysql> show slave status \G
```

Run IRP services on $IRPMASTER and $IRPSLAVE:

Listing 2.77: Starting IRP services and Frontend (OS with Systemd)

```
# systemctl start irp.target
```

Start services on $IRPMASTER first if the actions above took very long in order to shorten MySQL downtime.

Configure Failover using Wizard on $IRPMASTER

These steps should only be performed once SSH communication channel and MySQL Multi-Master irp database replication have been setup and are verified to work. Refer subsections of Failover Configuration above.

Run failover wizard:

Login into IRP's Frontend on $IRPMASTER node and run Configuration -> Setup wizard -> Setup Failover.

A valid failover license should be acquired before failover configuration settings become available.

Configure IRP failover:

Follow Setup Failover wizard steps and provide the required details. Refer Setup Failover wizard for details. Once your configuration changes are submitted IRP will validate configuration and if it is valid the configuration will be synced to $IRPSLAVE.

Only after this synchronization step takes place will $IRPSLAVE node know what is its role in this setup.

Apply configuration changes to edge routers:

Ensure designated probing IPs, BGP session(s) are also setup on edge routers. Refer to corresponding sections of this document for details.

Enable failover:

Use IRP's Frontend Configuration -> Global -> Failover to configure IRP failover on both nodes. Monitor both IRP nodes and ensure everything is within expectations.

It is recommended that after finishing the preparatory steps above both IRP master and slave nodes run with disabled failover for a short period of time (less than 1 hour) in order to verify that all IRP components and MySQL Multi-Master replication work as expected. Keep this time interval short to avoid a split-brain situation when the two nodes make contradictory decisions.

Synchronize RRD statistics to $IRPSLAVE

RRD statistics is collected by both IRP failover nodes and needs not be synchronized during normal operation. Still, when IRP failover nodes are setup at a large time interval between them it is recommended that RRD files are synchronized too. This will ensure that past statistics is available on both IRP nodes. During short downtimes of each of the nodes synchronization is less usefull but can be performed in order to cover the short downtime gaps in RRD based graphs.

Sample command to synchronize IRP's RRD statistics:

Listing 2.78: Synchronize RRD

```
root@IRPMASTER ~ # rsync -av /var/spool/irp/ $IRPSLAVE:/var/spool/irp
```

2.13.2 Re-purpose operational IRP node into an IRP failover slave

Sometimes an existing fully configured IRP node is designated to be re-purposed as a failover slave. Take the following actions to make a node as an IRP slave:

1. upgrade IRP to the version matching IRP failover master node
2. create a backup copy of your configuration
3. delete the following configuration files:
   - `/etc/noction/irp.conf`
   - `/etc/noction/exchanges.conf`
   - `/etc/noction/policies.conf`
   - `/etc/noction/inbound.conf`
4. proceed with configuration as detailed in Initial failover configuration

2.13.3 Re-purpose operational IRP failover slave into a new master

If a master node fails catastrophically and cannot be recovered an operational IRP slave can be re-purposed to become the new master. Once re-purposing is finished a new slave node can be configured as detailed in Initial failover configuration.

2.13.4 Recover prolonged node downtime or corrupted replication

Multi-Master replication used by IRP failover is able to cope with downtime of less than 24 hours. Replication will not be able to continue in case of prolonged downtime or corrupt replications. Recovery in this case might use either:

- new server: follow configuration steps as detailed in Initial failover configuration.
- same server: follow recovery steps below.

MySQL Multi-Master recovery prerequisites

Before proceeding with recovery of replication the following prerequisites must be met:

- Currently active IRP node is designated as MySQL sync 'origin'. This node currently stores reference configuration parameters and data. These will be synced to the node being recovered and we designate it as 'destination'.
- Recovery should be scheduled during non-peak hours.
- Recovery must finish before bgpd.db.timeout.withdraw (default 4h) expires. If recovery can not be completed in time it is required to start MySQL on the active node.

MySQL Multi-Master recovery procedure

It shall be noted that recovery closely follows actions described in and with the clarification that data and files from origin node target destination node (instead of IRP failover master files targeting IRP failover slave).

The steps are as follows:

1. destination: stop irp, mysqld
2. origin: sync /etc/noction/db.* to slave:/etc/noction/
3. origin: sync /root/.my.cnf to slave:/root/.my.cnf
4. origin: sync /var/lib/mysql/ to slave:/var/lib/mysql/
   exclude files:
   master.info relay-log.info -bin.* -relay.*

   wait until sync at (4) succeeds and continue with:
5. origin: stop irp (except bgpd), mysqld
6. origin: delete files master.info relay-log.info -bin.* -relay.*
7. origin: sync /var/lib/mysql/ to slave:/var/lib/mysql/
8. destination: start mysqld and check /var/log/mysqld.log for errors
9. origin: start mysqld and check /var/log/mysqld.log for errors
10. origin: run CHANGE MASTER TO from the /usr/share/doc/irp/changemasterto template
11. destination: run CHANGE MASTER TO from the /usr/share/doc/irp/changemasterto template
12. destination: show slave status \G
13. origin: show slave status \G
14. origin: start IRP (bgpd should be already running)
15. destination: start IRP

Chapter 3

Using IRP

3.1 Getting started

The **GMI Frontend** represents a web interface with a comprehensive set of reports, graphs and diagnostic information which can reflect the current and historical network state, as well as the network optimization benefits delivered by IRP instances.

3.1.1 Accessing the system

The system frontend is accessible over HTTPS. GMI's Frontend can be accessed using any major browser, via the server IP address.

3.1.2 Creating GMI user

When opening the GMI Frontend's for the very first time, a login form will pop up asking you to create a user.

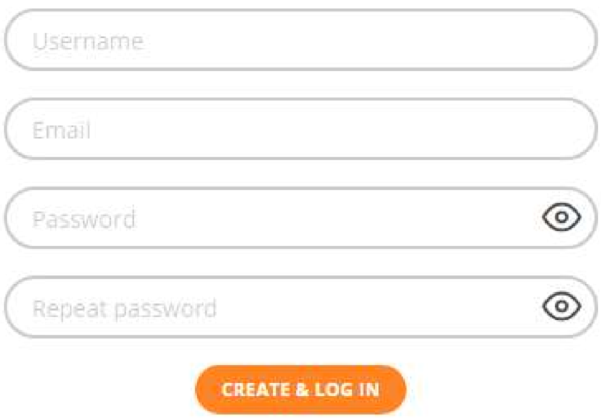

Figure 3.1.1: Creating initial user in GMI

3.1.3 Auto-Registration of Local IRP Instances into GMI

When installed on the same server, IRP instances automatically register with GMI. The GMI authorization form then offers the choice to complete the registration or add other IRP instances.

3.1.4 Registering IRP instances using root passwords

To register an IRP instance in GMI, simply enter the IRP server's root password in the GMI authorization form and click 'Finish'.

The root password will be securely transmitted to the IRP instance over SSL, exclusively for authorization purposes. It will not be stored or utilized for any other activities beyond authorization..

3.1.5 Registering IRP instances using tokens

Creating a GMI token authorizes the Global Management Interface (GMI) to access an IRP instance. Here's an outline of the registration process:

- Access the IRP Instance CLI: First, access the Command Line Interface (CLI) of your IRP instance, either through a direct console connection or remotely via a terminal application.
- Generate a Token: Use the IRP Management Tool (Irpmng) to generate a GMI token. This token acts as a secure authorization key for GMI access (for detailed instructions follow Creating GMI Tokens section.
- Save the Token: It's important to save the GMI token immediately after generation, as it cannot be retrieved once lost. Keeping the token secure ensures it's available for future authorization of the IRP instance into GMI.
- Register the IRP Instance: In the GMI authorization form, enter the token and the IP address of the IRP instance to complete the registration.

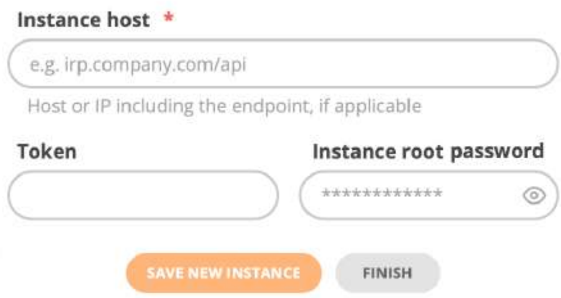

Figure 3.1.2: GMI authorization form

3.1.6 Creating GMI Tokens

GMI tokens can be generated using the IRP Management Tool (Irpmng), accessible through the IRP instance CLI.

Here's a step-by-step guide:

- Access GMI Instance CLI: Connect to the GMI instance Command Line Interface (CLI) either through a direct console connection or remotely via a terminal.
- Use GMI CLI Tool: Use the GMI CLI tool to obtain the GMI instance's unique identifier by executing the following command:

```
# gmi-cli showID
GMI UID: 7cdbc6c8a81e469226ffb174a2bf399d
```

- Access IRP Instance CLI: Connect to the IRP instance Command Line Interface (CLI) either through a direct console connection or remotely via a terminal.
- Use IRP Management Tool: Create token using the command below

```
# irpmng gmi add --shortname <SHORTNAME> --admin true --uid <GMIUID>
GMI token IRP[1] has been added as: Admin, Enabled
Write down the token value as it cannot be retrieved later:
b84edc081481b339edcc1d51c96e2668663f894848ffa9556ed0c243c1611460
```

<SHORTNAME> - This is the desired short name for the IRP instance.

<GMIUID> - This is the GMI unique identifier you retrieved earlier.

⚠ Save the Token: It's imperative to save the GMI token immediately after generation, as it cannot be retrieved once the command window is closed or if the token is lost. This token is essential for future authorizations and registrations of the IRP instance in GMI.

- Use the GMI authorization form to complete the IRP instance registration

3.1.7 Using the IRP management tool for token management

The IRP Management Tool (Irpmng) provides comprehensive options for managing GMI tokens, ensuring flexibility and control over token lifecycle and access rights.

- Addition: This feature enables the generation of new GMI tokens. It's useful when new instances need to be registered or additional access needs to be granted.
- Modification: This allows for adjustments to be made to existing GMI tokens. Modifications can include changes to the token's privilege level or its active state, accommodating evolving security requirements or operational changes.
- Deletion: To maintain optimal security and organization, outdated or unnecessary GMI tokens can be removed from the system. Regularly pruning tokens reduces potential attack vectors and keeps the token inventory manageable.
- Viewing: Users have the capability to inspect the details of existing GMI tokens. Information such as token short names, privilege levels, associated GMI UIDs, and the last access timestamp can be reviewed. This is crucial for auditing and tracking token usage.

The general syntax for executing commands related to GMI token management using the Irpmng module is as follows:

```
# irpmng gmi <command> <options>
COMMAND:
      add - Adds a new token with specified parameters
      del - Provides delete operation
      list - List all existing tokens and their properties
      set - Patches specific token properties for the given GMI UID
      show - Displays a token in vertically formatted form for the given
         GMI UID
      help - print help information
OPTIONS:
      -t, --token <token> - Desired token value
      -s, --shortname <shortname> - Desired token shortname
      -u, --uid <gmi_uid> - GMI unique identifier (UID)
      -a, --admin <admin> - Token administrative privileges [possible
         values: true, false]
      -e, --enabled <enabled> - Token status enabled or disabled [possible
         values: true, false]
```

Add a new GMI token:

```
# irpmng gmi add --token <TOKEN> --admin=true --uid <GMIUID>
GMI token IRP[1] has been added as: Admin, Enabled
Write down the token value as it cannot be retrieved later:
ef4c474e3d0676d0de0fb584d3e9bc87784285483df9094d75bb555e6c82f093
```

Remove a GMI token:

⚠ Deleting a GMI token will result in all data being lost including per-user settings, access tokens and notification subscriptions

```
# irpmng gmi del <GMIUID>
NOTE: The operation is irreversible.
GMI token removal will remove all the user tokens created by that GMI
    instance
Do you want to delete token IRP[1] (Y/n)?
GMI token IRP[1] has been deleted
```

Show a GMI token:

```
# irpmng gmi show <GMIUID>
Id: 1 Rights: Admin
State: Enabled
Last access time: -
Shortname: IRP
GMI UID: d5b457501d230523a8a466babeb8d67a
```

List GMI tokens:

```
# irpmng gmi list
 Id | Rights | State | Last access time | Shortname
  1 | Admin | Enabled | 2024-03-05 14:58:50 | IRP-America
  2 | User | Disabled | | IRP-Europe
```

Set a GMI token:

```
# irpmng gmi set <GMIUID> --admin false --enabled false
GMI token IRP[1] has been changed admin rights revoked state changed to
    disabled
```

Additionally, for improved security, it's advisable to let the system automatically generate the token. This method ensures the creation of robust, complex tokens less susceptible to compromise.

To facilitate user interaction with the IRP Management Tool, the command "irpmng gmi help" is available. Executing this command presents a detailed overview of all the commands and options within the tool, aiding users in effectively utilizing its features.

Leveraging these guidelines and the capabilities of the Irpmng tool, administrators can adeptly manage GMI tokens. This careful management contributes to secure and efficient access to the Global Management Interface, ensuring the integrity and security of IRP instance interactions.

3.1.8 Utilizing the GMI command line interface

The GMI command line interface proposes several configuration options:

- Show the GMI UID
- Authorization of IRP instance into GMI using a created in advance token
- Addition of users with desired privilege levels

```
# gmi-cli <command> <options>
COMMAND:
      showID  - Displays a GMI unique identifier (UID)
      addInstance - Addition of IRP instance
      addUser  - Addition of GMI user with their parameters
      upgradeConfig - Upgrade the configuration
      help - print help information
OPTIONS:
      host <hostname/IP address> - IRP instance hostname or IP address
      token <GMI Token> - GMI token
      shortName <shortname> - IRP instance shortname (optional)
      username <username>   - User username
      password <password> - User password
      email <email> - User email address
      role <role> - User privileges [possible values: admin, manager, user]
```

For example, to add a new IRP instance:

```
# gmi-cli addInstance <HOSTNAME/IP ADDRESS> <GMI TOKEN> <SHORTNAME>
Instance IRP at https://127.0.0.1 is being added
connected to server!
Response: Added instance successfully!
disconnected from server
```

For example, to add a new GMI user:

```
# gmi-cli addUser <USERNAME> <PASSWORD> <EMAIL> <ROLE>
addUser: IRP-ADMIN 0523a8a466 admin@example.com admin
connected to server!
Response: Added user successfully!
disconnected from server
```

To assist users in navigating the available commands and options, the "gmi-cli help" command provides a comprehensive list of all available commands and their respective options.

3.1.9 Login into GMI

On consecutive visits the GMI username and password need to be provided to login.

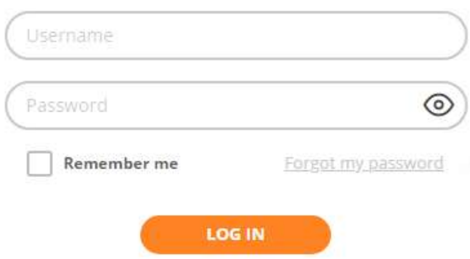

Figure 3.1.3: GMI Frontend login form

Frontend structure

After a successful login, the system's dashboard will be displayed. The Frontend has several components that operate together and allow a regular user or administrator to access various reports, graphs, or configuration pages.

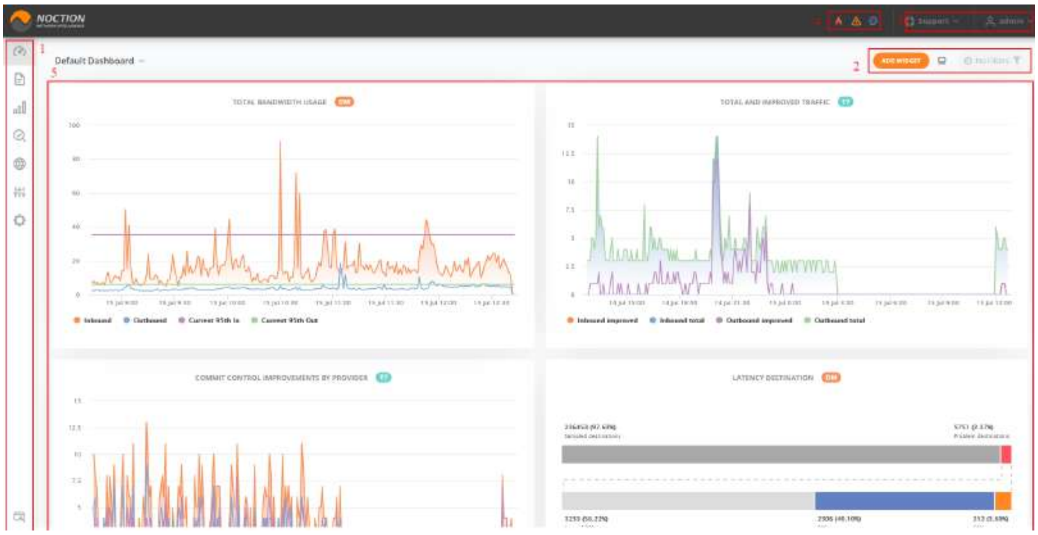

Figure 3.1.4: Frontend Sections

1. The main left sidebar allows quick access to any part of the system via its menu items.
2. A variation of controls appears here in dashboards, reports and graphs depending on the actual view.
   Clicking the timestamp/filter icon opens up the menu on the right side (See section 3.1.10).
   Its contents depend on the current page, and include the time selector as well as specific filters, as explained in the next section.
3. Various links to Support, User Guide, API Documentation and the User account administration are located in the top right corner of the Frontend.
4. The status bar allows one to access system events and notifications (See the Events section).
5. Main dashboard area.
   Main dashboard area is editable, widgets can be added, removed or rearranged.

Keyboard shortcuts

- Dashboard - SHIFT+D
- Reports - SHIFT+R
- Graphs - SHIFT+G
- Inbound - SHIFT+I
- Policies - SHIFT+J
- Troubleshooting - SHIFT+T
- Threat Mitigation - SHIFT+B
- Global Commit - SHIFT+M
- Configuration - SHIFT+C
- Management - SHIFT+S
- Search - Ctrl + K
- Instance sidebar - SHIFT+E

3.1.10 Right Sidebar

All dashboards, reports, graphs and events have a sidebar that opens up on click of the timestamp/filter icon at the right side of the screen. Depending on the current page, the sidebar has a different set of controls, allowing users to select a desired timeframe or to filter any report/graph results.

On graphs, reports, events page and dashboards, the time period which the information is being provided for, can be selected by using the "Timestamp" button. An intuitive control will open, and the desired time period can be selected.

Figure 3.1.5: Time period selection

In the case of a report, graph or events page, custom filters can be applied, so that only the specific information is displayed.

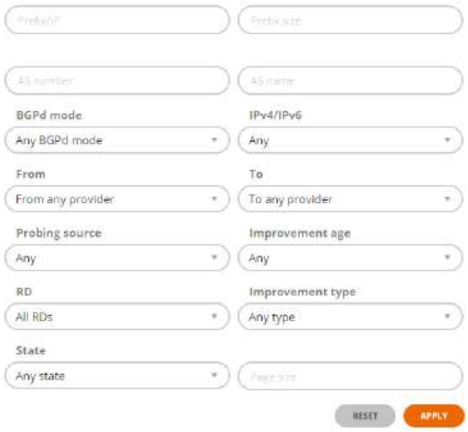

Figure 3.1.6: Report filters

3.1.11 IRP Instance Performance Stats

Clicking a particular IRP short-name button, opens the right hand side panel with instance-specific information:

- Current amount of inbound and outbound traffic as well as a quick bandwidth reference graph for the last 5 minutes.
- Configured maximum number of improvements (Max IPv6 Improvements will be shown if IPv6 is enabled)
- Number of Routing Policies improvements
- Number of improvements performed today
- The amount of traffic improved today
- Percentage of prefixes improved today
- Number of operating iBGP sessions

- Number of announcements to the edge router
- Announcements to each from configured routers
- Number of BGP sessions with the providers
- Number of operating BGP sessions with the providers

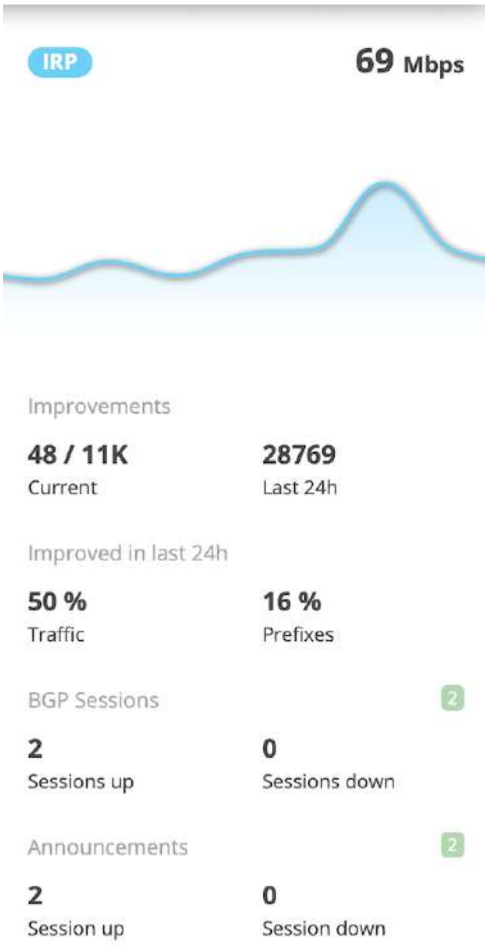

Figure 3.1.7: IRP Instance Performance Stats Quick Reference

See also: GMI Dashboards.

3.1.12 Global Search

GMI frontend allows to search for historical improvements and the AS Path problems information on prefixes, AS Numbers, AS Names, etc. This function can be accessed by clicking the Search icon in the top bar.

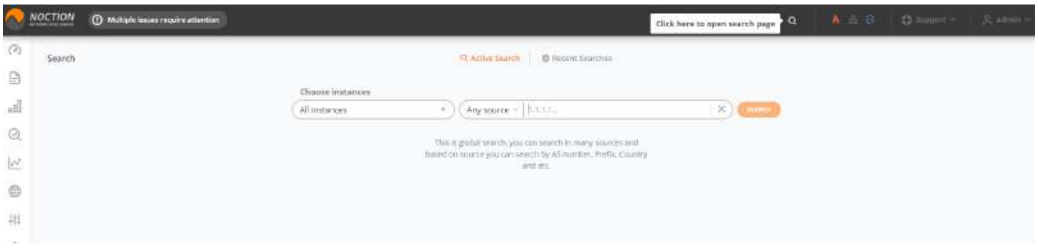

Figure 3.1.8: Global Search

Choose all instances or the IRP instance of interest, enter the needed prefix, AS number or AS name in the search box, then press the SEARCH button.

When displaying the complete results, specific information for a prefix can be obtained or it can be probed, by clicking the corresponding icons next to each result.

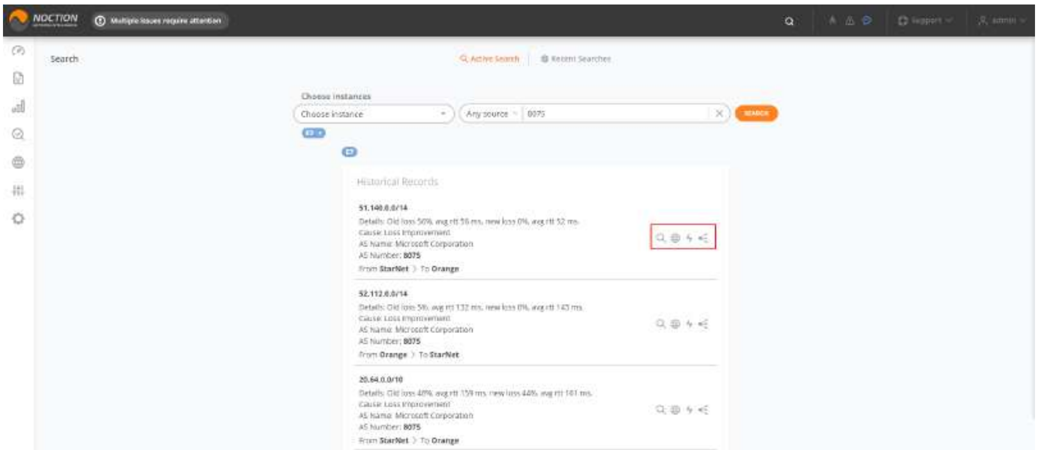

Figure 3.1.9: Global Search results

Visit the Historic Searches tab to see the list of recent search queries.

Figure 3.1.10: Recent Searches

3.1.13 Warning bars

The warning bars are shown at the top of the GMI frontend in case the attention should be drawn to operational/performance issues, some specific IRP instance configuration settings or license/payment issues.

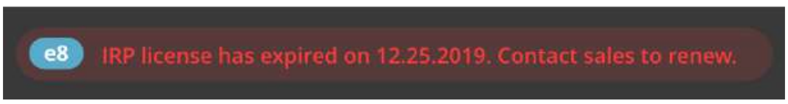

Figure 3.1.11: IRP instance license expiration warning example

3.1.14 API documentation

The Swagger-formatted API documentation is conveniently accessible at the upper right-hand corner of the dashboard, located within the "Support" tab. This comprehensive resource furnishes an exhaustive catalogue of all extant API endpoints within the IRP system, along with their respective architectural delineations

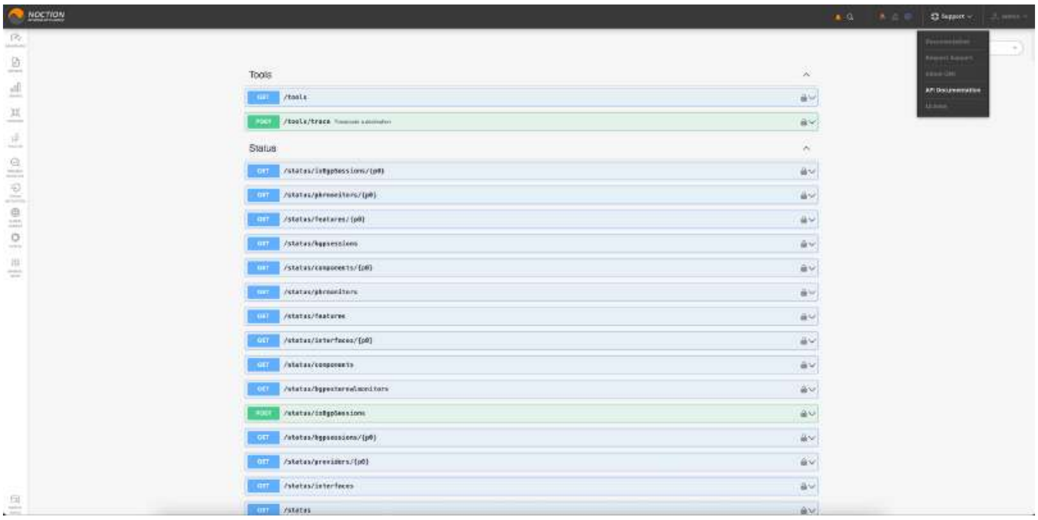

Figure 3.1.12: API Documentation

3.2 Wizards

Wizards can be accessed from Configuration→ Setup Wizards or from Configuration → Frontend → Configuration Wizards.

3.2.1 Initial Setup

IRP instance initial setup wizard includes the following steps:

1. Specifying Infrastructure IP addresses
2. Providing the list of analyzed networks
3. Configuring Span Collector
4. Configuring Flow Collector
5. Selecting IRP's Improvement mode
6. Setup the management interface
7. Indicate interfaces that IRP uses for active probing

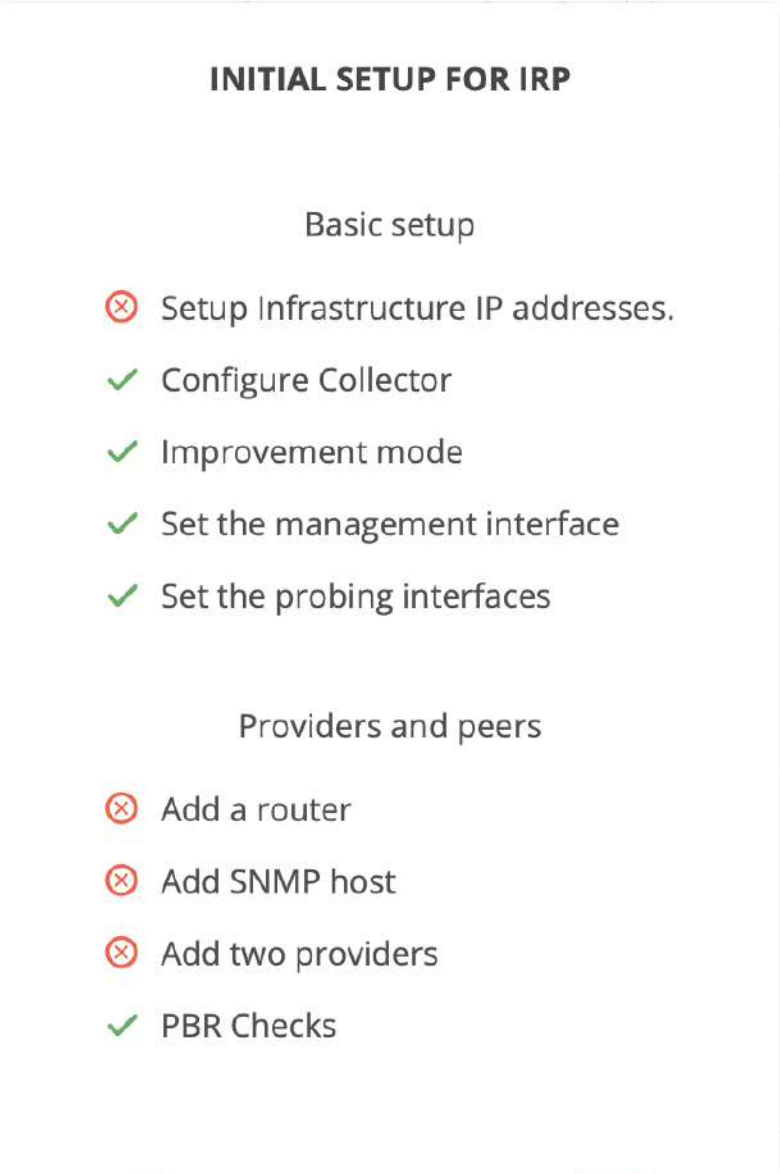

Figure 3.2.1: Configuration editor: Initial Setup

- Basic Setup
  - Setup Infrastructure IP addresses (explorer.infra_ips)

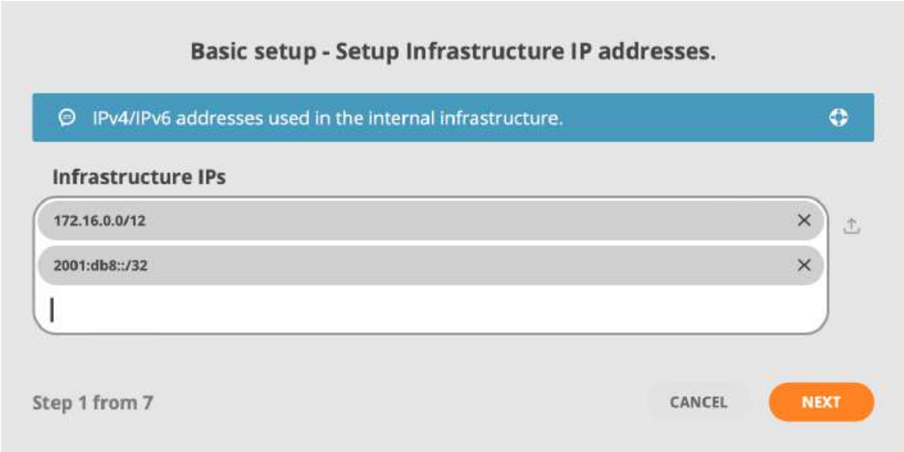

Figure 3.2.2: Configuration editor: Setup Infrastructure IP addresses

– Setup Analyzed networks (collector.ournets)

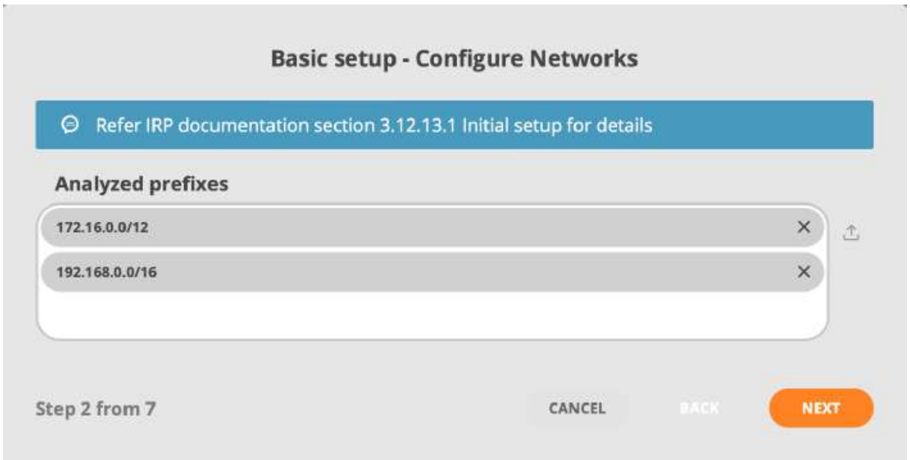

Figure 3.2.3: Configuration editor: Setup Analyzed networks

– Configure Collector:
  * Irpspand
    · Irpspand enable/disable (collector.span.enabled)
    · Irpspand interfaces (collector.span.interfaces)
    · Mindelay status (collector.span.min_delay)
    · Mindelay probing queue slots (collector.span.min_delay.probing_queue_size)

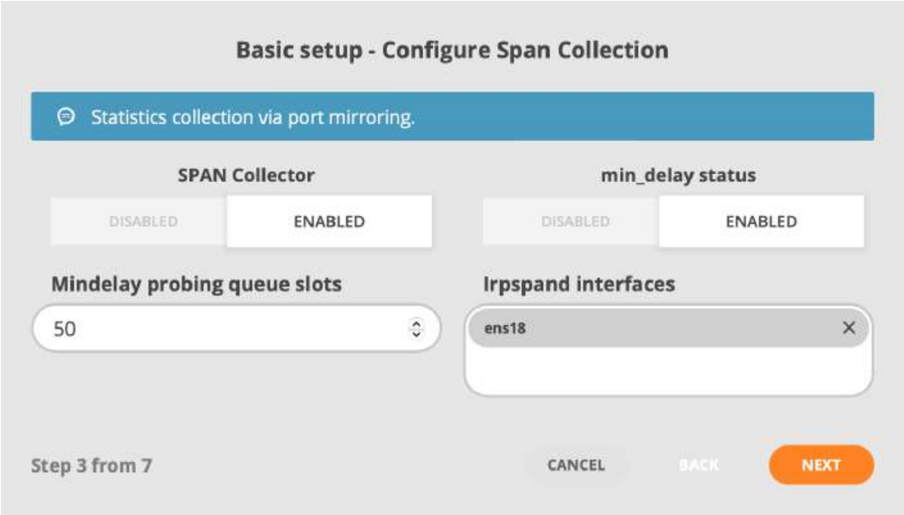

Figure 3.2.4: Configuration editor: Configure SPAN Collector

* Irpflowd
  · Irpflowd enable/disable(collector.flow.enabled)
  · NetFlow UDP port(collector.flow.listen.nf)
  · sFlow UDP port(collector.flow.listen.sf)
  · Flow Sources(collector.flow.sources)

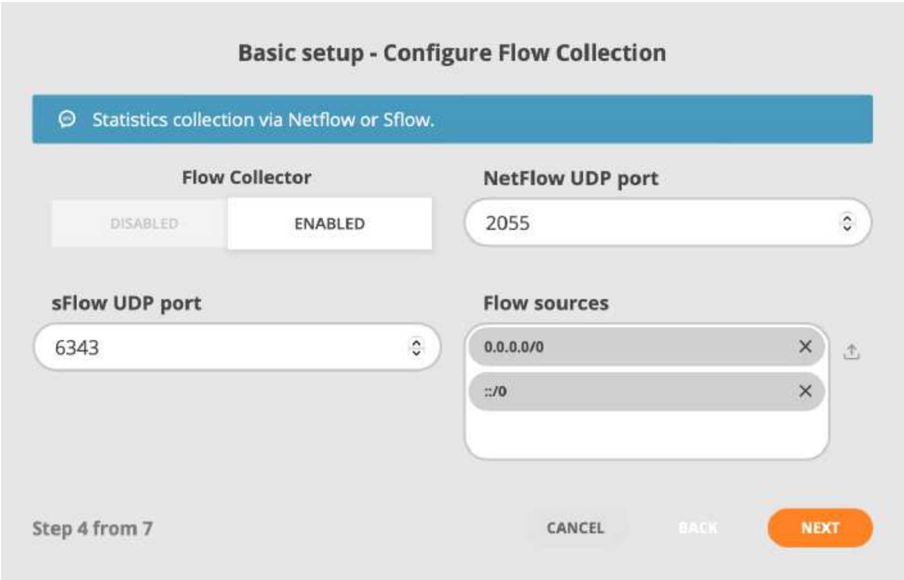

Figure 3.2.5: Configuration editor: Configure Flow Collector

– Improvement mode (global.improve_mode)

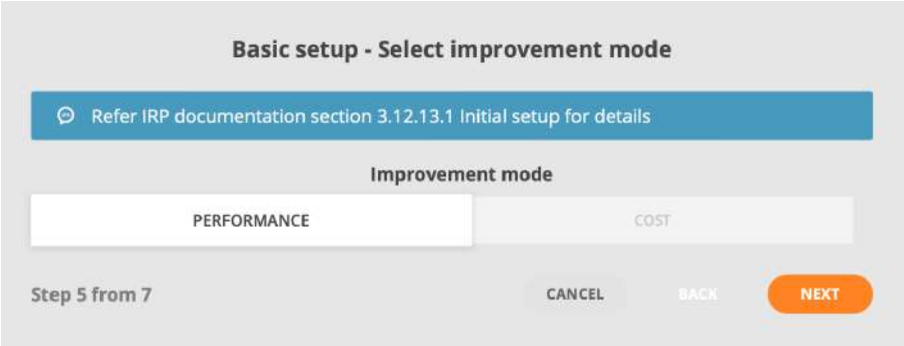

Figure 3.2.6: Configuration editor: Improvement mode

– Set the managements interface (global.master_management_interface)

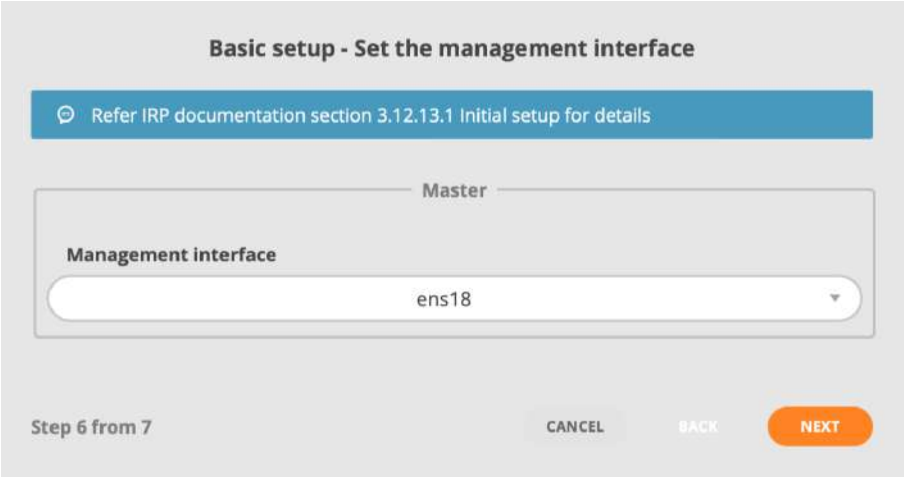

Figure 3.2.7: Configuration editor: Management Interface

– Set the probing interface (global.master_probing_interface)

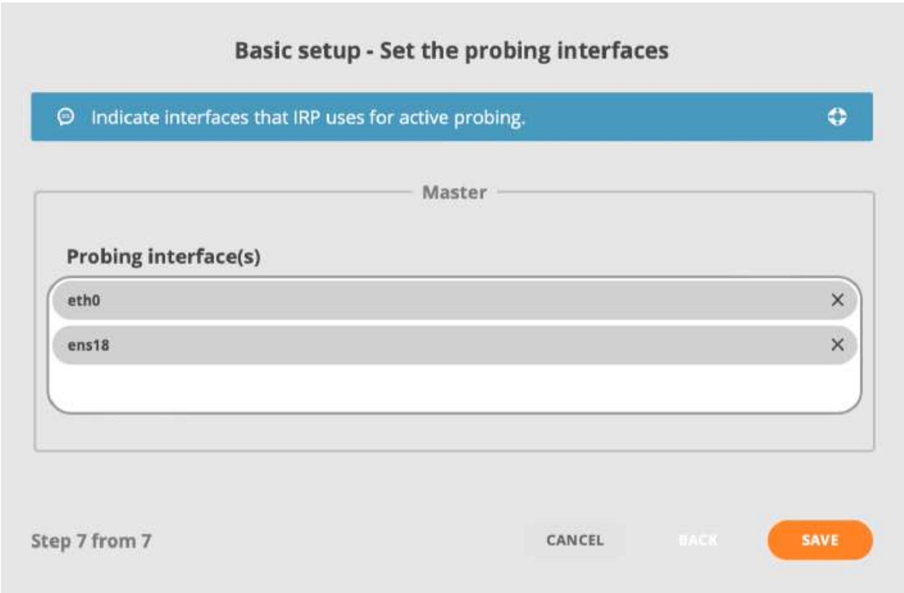

Figure 3.2.8: Configuration editor: Probing Interface

• Providers Setup

– Add a router, see Add Router
– Add two providers, see Add Provider

3.2.1.1 Add Router

IRP communicates improvements to your edge routers. The Add Router wizard guides you through the router configuration with the following steps:

1. Identify the router and its AS
2. Set up the Router IP address for the BGP session
3. Define BGP announcement attributes to distinguish and prioritize IRP improvements on this router

Check below the corresponding wizard steps:

- Router name assigned within IRP for easy identification
- Autonomous System number of the network (bgpd.peer.X.as)

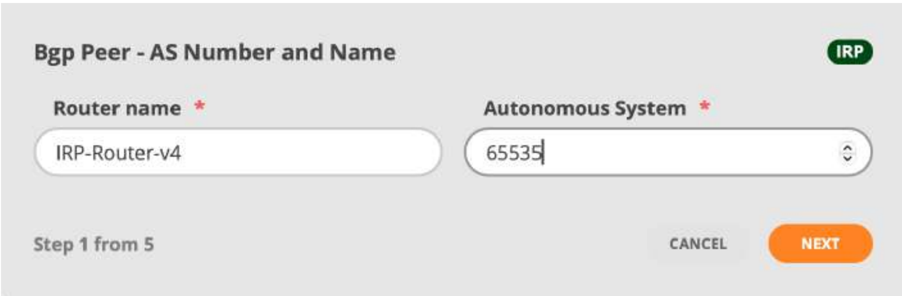

Figure 3.2.9: Configuration editor: Router name and AS

- Router General parameters (IPv4)
  – IRP's IPv4 address (bgpd.peer.X.master_our_ip)
  – Router IPv4 address (bgpd.peer.X.master_peer_ip)
  – Announced Improvement LocalPref value (bgpd.peer.X.master_localpref)

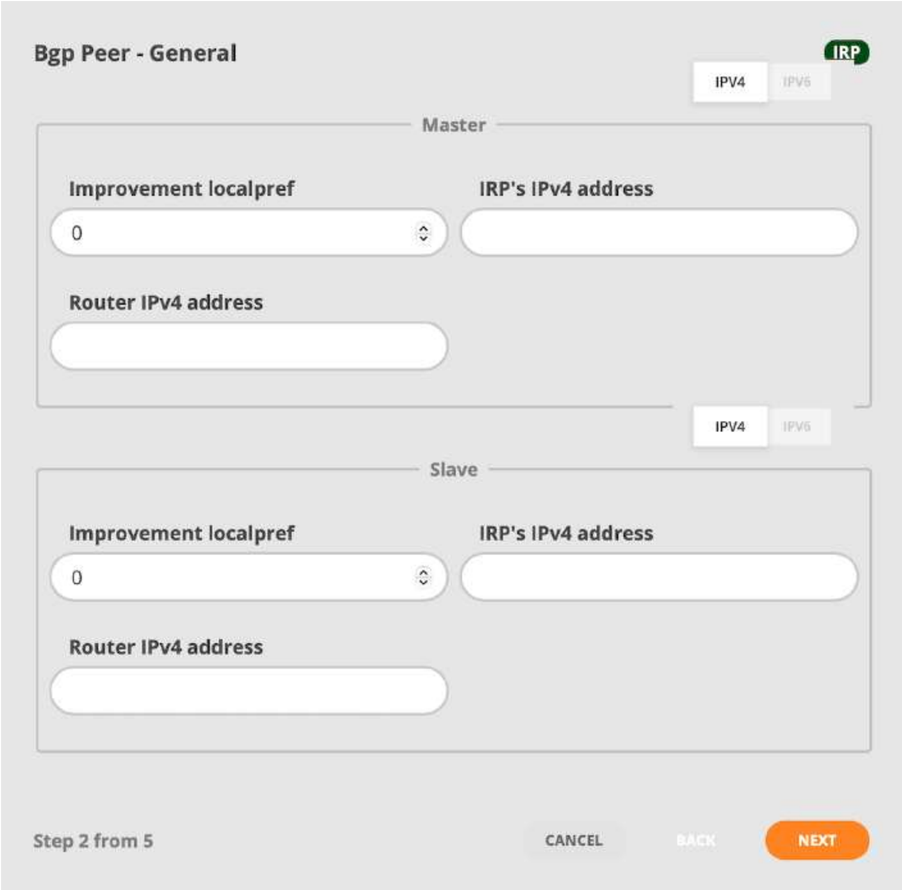

Figure 3.2.10: Configuration editor: Router IPv4 addresses

- Router General parameters (IPv6)
  - IRP's IPv6 address (bgpd.peer.X.master_our_ipv6)
  - Router IPv6 address (bgpd.peer.X.master_peer_ipv6)
  - Announced Improvement LocalPref value (bgpd.peer.X.master_localpref)

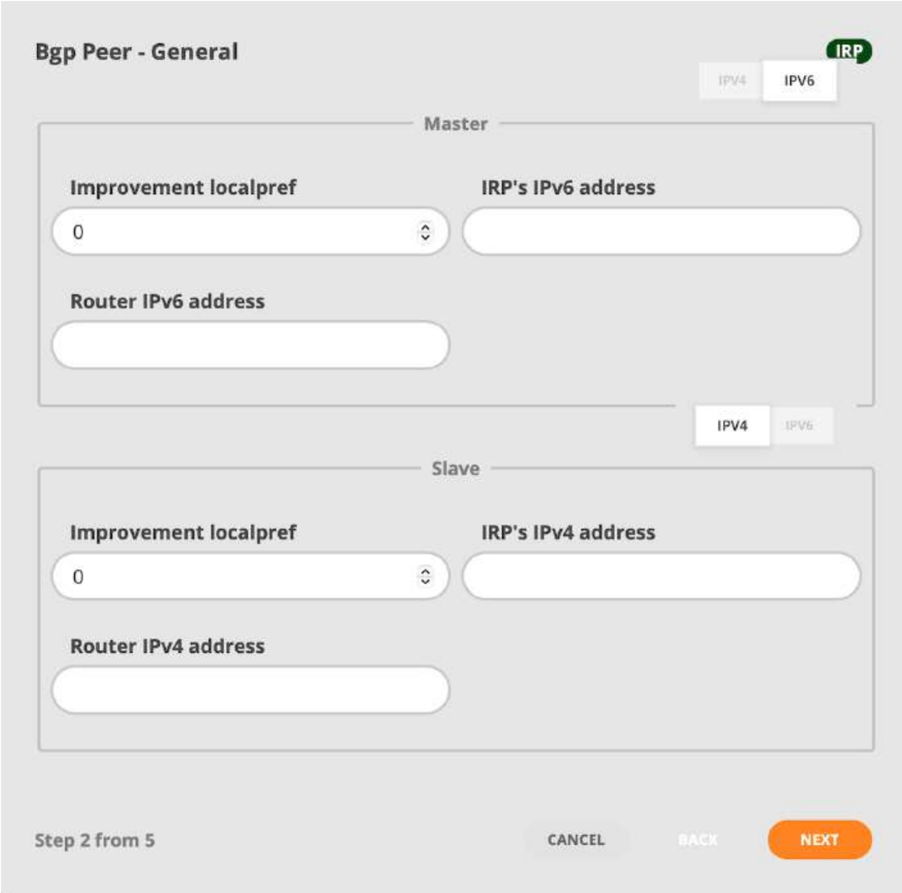

Figure 3.2.11: Configuration editor: Router IPv6 addresses

- Router Advanced parameters
  - Flowspec status (bgpd.peer.X.flowspec)
  - Flowspec redirect type(bgpd.peer.X.flowspec.redirect_type)
  - Keepalive interval (sec) (bgpd.peer.X.keepalive)
  - BGP session password (bgpd.peer.X.master_password)
  - Improvement communities (bgpd.peer.X.master_communities)
  - BGP Router ID (bgpd.peer.X.master_router_id)
  - FlowSpec redirect type (bgpd.peer.X.flowspec.redirect_type)

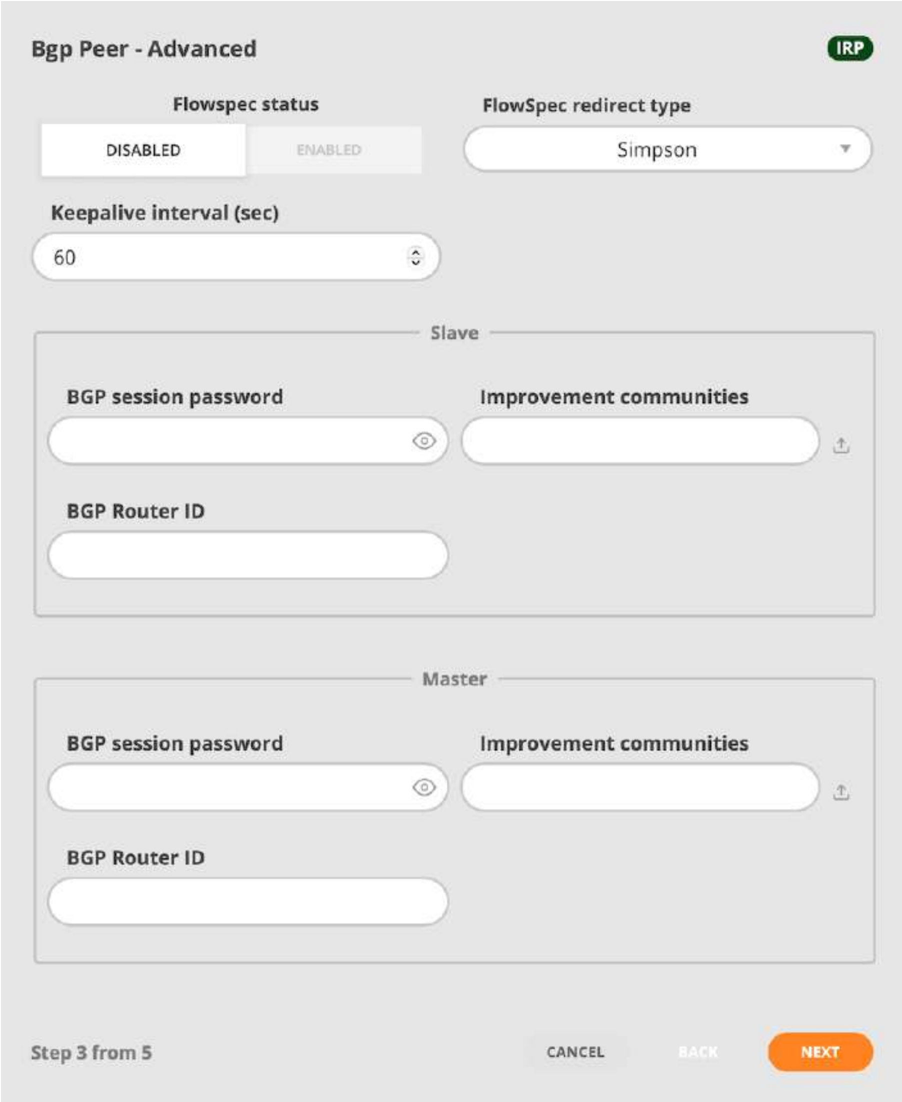

Figure 3.2.12: Configuration editor: Advanced parameters

- Router Inbound parameters
  - Local IPv4/IPv6 inbound next_hop (bgpd.peer.X.inbound.ipv4.next_hop, bgpd.peer.X.inbound.ipv6.next_ho
  - Transit SNMP hosts (bgpd.peer.X.transit.snmp)
  - Transiting traffic (bgpd.peer.X.transit.status)
  - Announced inbound localpref value (bgpd.peer.X.inbound.master_localpref) (bgpd.peer.X.inbound.slave_local

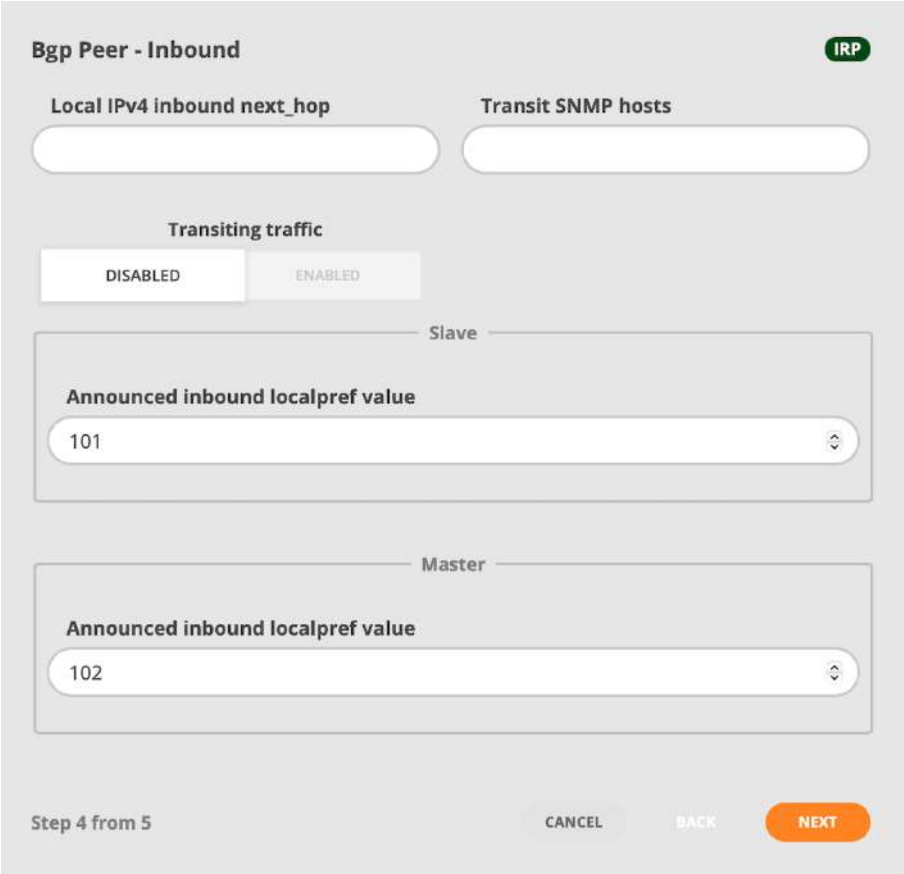

Figure 3.2.13: Configuration editor: Inbound attributes

- Router Blackholing parameters
  - Announced Blackholing LocalPref value (bgpd.peer.X.blackholing.localpref)
  - Blackholing IPv4 next_hop (bgpd.peer.X.blackholing.ipv4.next_hop)
  - Blackholing IPv6 next_hop (bgpd.peer.X.blackholing.ipv6.next_hop)

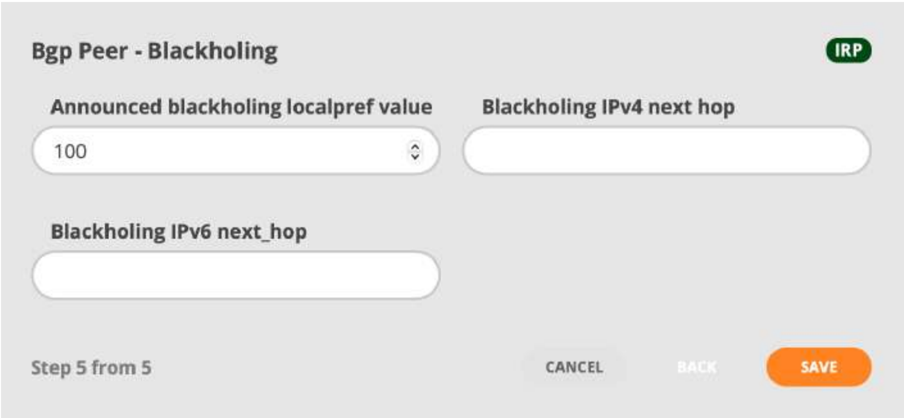

Figure 3.2.14: Configuration editor: Inbound attributes

3.2.1.2 Add Provider

The wizard Add Provider covers the following:

1. Router - Identify the router and routing domain where the provider interconnects with your network

2. Provider - Specify what is a provider ASN, short name and description
3. IP addresses - Specify and assign provider network addresses that IRP will use
4. Commit Control - Optionally set provider usage thresholds to be used for Commit Control
5. SNMP host - Specify the SNMP host for this provider
6. External monitor - Indicate if an External monitor is used and designated external IP addresses used to verify this provider link status
7. Internal monitor - Indicate if an Internal monitor based on BGP session with provider will be used and the corresponding SNMP resource
8. Pre-checks - Validate given provider parameters before submitting them

- Choose a provider type (Transit, Partial or Exchange) (peer.X.type) images/wizards/AddProviders/

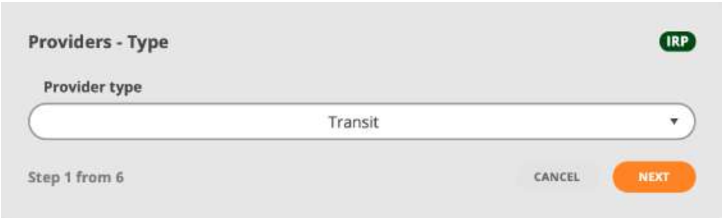

Figure 3.2.15: Configuration editor: Choose a provider type

- Choose a Router (peer.X.bgp_peer)
- Provider/link name:
  - Provider short name (peer.X.shortname)
  - Provider description (peer.X.description)

Figure 3.2.16: Configuration editor: Provider name

- Provider IPv4 addresses
  - IPv4 diagnostic hop (peer.X.ipv4.diag_hop)
  - Probing IPv4 address (peer.X.ipv4.master_probing)
  - Router next-hop IPv4 address (peer.X.ipv4.next_hop)
  - Router ASN for IPv4 (peer.X.ipv4.next_hop_as)

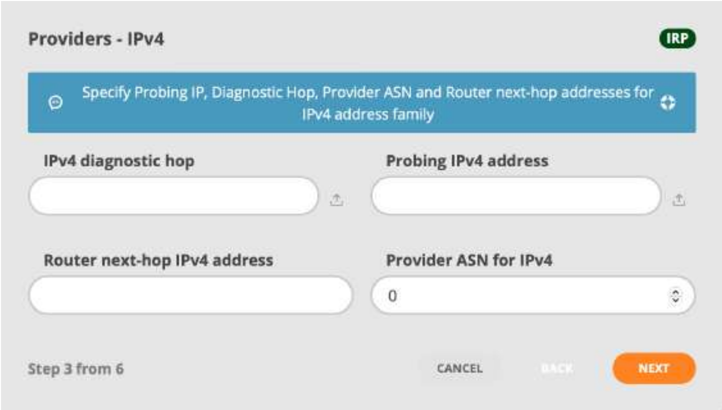

Figure 3.2.17: Configuration editor: Provider IPv4 address

- Provider IPv6 addresses
  - IPv6 diagnostic hop (peer.X.ipv6.diag_hop)
  - Probing IPv6 address (peer.X.ipv6.master_probing)
  - Router nex-hop IPv6 address (peer.X.ipv6.next_hop)
  - Router ASN for IPv6 (peer.X.ipv6.next_hop_as)

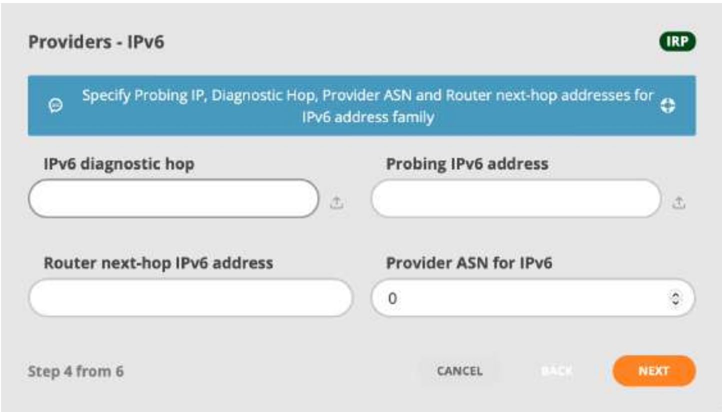

Figure 3.2.18: Configuration editor: Provider IPv6 addresses

- Provider Commit Control
  - Provider 95th percentile (peer.X.95th)
  - Commit Control status for this provider (peer.X.cc_disable)
  - Allow Improvements within provider group members (peer.X.improve_in_group)

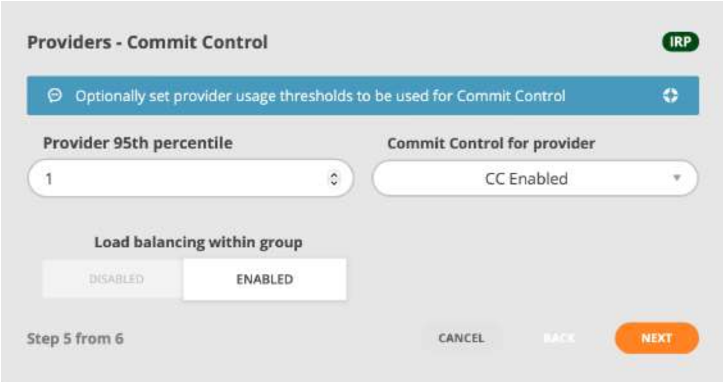

Figure 3.2.19: Configuration editor: Provider Commit Control

- Provider Monitoring setup
  - SNMP host settings (SNMP Host)
  - Provider SNMP interfaces (peer.X.snmp.interfaces)

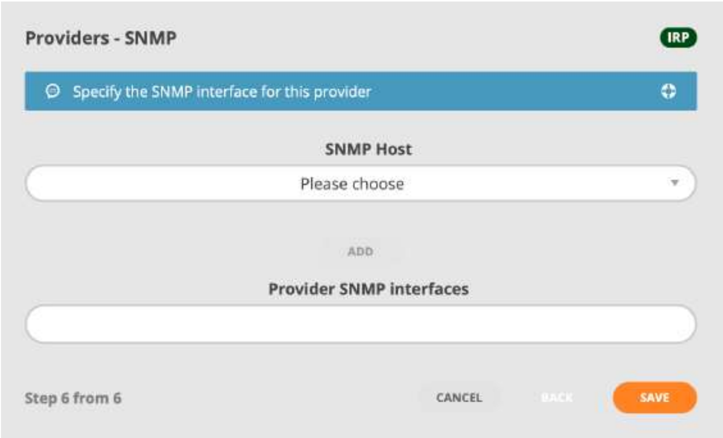

Figure 3.2.20: Configuration editor: Provider SNMP

3.2.1.3 Setup Commit Control

Step 1: Enable Commit Control Providers that should maintain a target 95th limit are identified and the exact limits are set. Link upper limit is recommended to set as 80-90% of the maximum interface capacity

- Commit control enables/disables the feature for a specific provider(peer.X.cc_disable)
- 95th target specifies the contracted bandwidth usage target(peer.X.95th)
- Link upper limit specifies the maximum allowed bandwidth on the link in Mbps (peer.X.limit_load).

Figure 3.2.21: Configuration editor: Commit Control basic setup

Step 2: Provider precedence Precedence indicates how important it is to keep the commit levels for different providers. Traffic will be unloaded to the provider with least precedence in situations when all other providers are overloaded. If two or more providers have equal precedence they will form groups. Move the slider or type in the desired values to specify the order of providers from most important to least important.

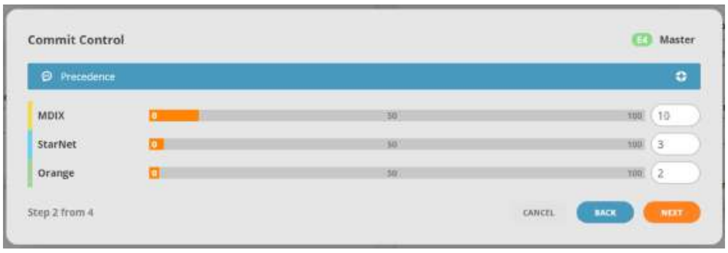

Figure 3.2.22: Configuration editor: Commit Control provider precedence setup

Step 3: Groups configuration Groups are formed by a set of providers (as configured in IRP) that share some common characteristics, for example a redundant link or a virtual link consisting of multiple physical links towards the same upstream provider. Depending on customer's needs, groups can be configured to permit or forbid performance rerouting from one provider in the group towards another. If providers are grouped they will be assigned the same precedence.

Groups are optional and can be skipped.

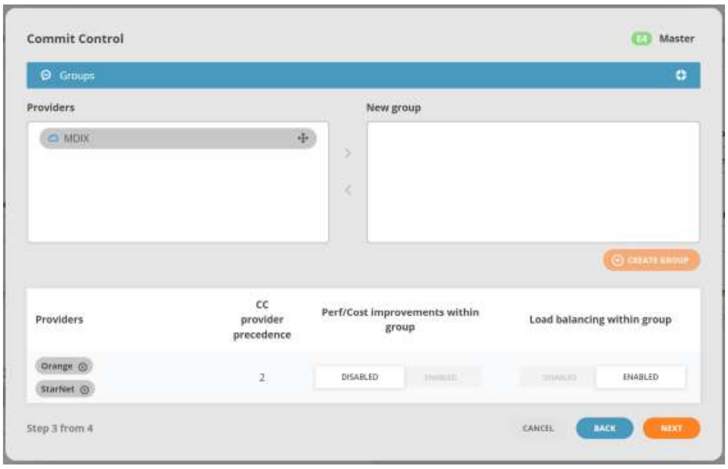

Figure 3.2.23: Configuration editor: Commit Control groups setup

Step 4: Commit Control Overview The last step of the wizard is there to review the configuration. When submitted the configuration changes will be validated and applied if correct.

- Provider name (peer.X.shortname)
- Commit control status for a specific provider (peer.X.cc_disable)
- Configured 95th (peer.X.95th)
- Upper limit (peer.X.limit_load)
- Load balancing for groups (peer.X.group_loadbalance)
- Improvements within group (peer.X.improve_in_group)
- Precedence (peer.X.precedence)

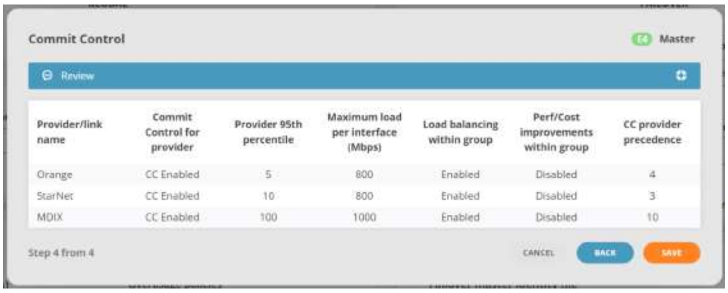

Figure 3.2.24: Configuration editor: Commit Control Provider overall

3.2.1.4 Setup Failover wizard

Step 1: Enable failover Failover configuration has a series of mandatory global settings before enabling it.

- Failover status (global.failover_role)
- Slave IPv4 address (global.failover_slave.ip)
- Slave IPv6 address (global.failover_slave.ipv6)
- Slave SSH port (global.failover_slave.port)
- Failover timer in seconds (global.failover_timer_fail)
- Failback timer in seconds (global.failover_timer_failback)

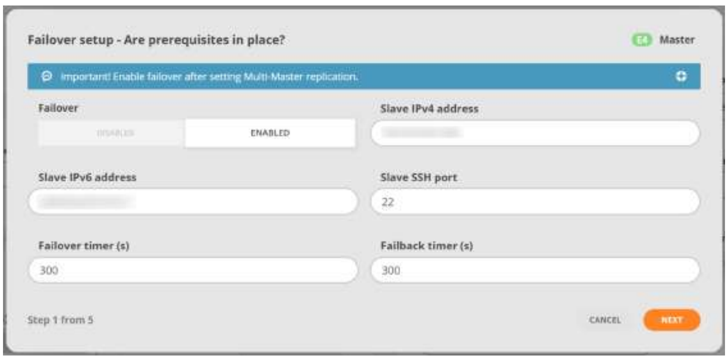

Figure 3.2.25: Configuration editor: Failover setup

Step 2: Probing IP configuration Probing IPs must be assigned to all configured providers for both master and slave nodes (peer.X.ipv4.master_probing, peer.X.ipv4.slave_probing, peer.X.ipv6.master_probing, peer.X.ipv6.slave_probing).

Figure 3.2.26: Configuration editor: Failover probing IPs for providers

Step 3: Router BGP session settings Master and slave nodes of a failover configuration establish BGP sessions with all configured routers. The following parameters are required:

- Local IP address for both master and slave (bgpd.peer.X.master_our_ip, bgpd.peer.X.slave_our_ip, bgpd.peer.X.master_our_ipv6, bgpd.peer.X.slave_our_ipv6)
- Router IP address for both master and slave (bgpd.peer.X.master_peer_ip, bgpd.peer.X.master_peer_ipv6, bgpd.peer.X.slave_peer_ip, bgpd.peer.X.slave_peer_ipv6)
- BGP session password for master and/or slave if any (bgpd.peer.X.master_password, bgpd.peer.X.slave_password)

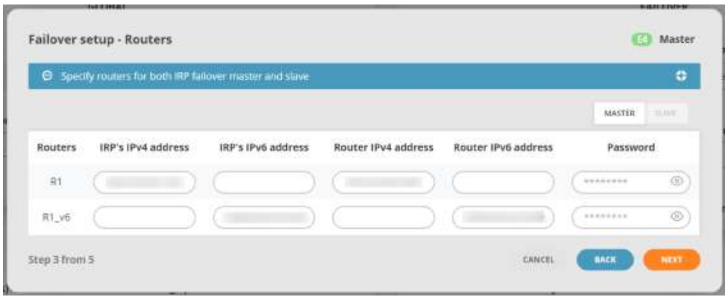

Figure 3.2.27: Configuration editor: Failover BGP session settings

Step 4: BGP LocalPref and Communities Master and slave nodes apply BGP attributes to announcements:

- LocalPref for master and slave (bgpd.peer.X.master_localpref, bgpd.peer.X.slave_localpref)
- Communities for master and slave (bgpd.peer.X.master_communities, bgpd.peer.X.slave_communities)

Master's LocalPref value must be greater than slave's LocalPref.

Figure 3.2.28: Configuration editor: BGP announcement attributes

Step 6: Probing interfaces Failover requires configuration of Management and Probing interfaces for both master and slave nodes (global.master_probing_interface, global.slave_probing_interface). Toggle between master and slave settings to specify values for both nodes.

Figure 3.2.29: Configuration editor: Probing and management and interfaces

3.3 GMI Dashboards

GMI dashboards are the specific sets of interactive visualizations, designed for quick analysis of the IRP instances performance and informational awareness.

Dashboards consist of widgets which contain a reduced version of any given report or a graph. These can be added, edited, deleted or modified as you like. GMI allows users to set up multiple dashboards.

To see a list of existing dashboards, go to "Default Dashboard > ALL".

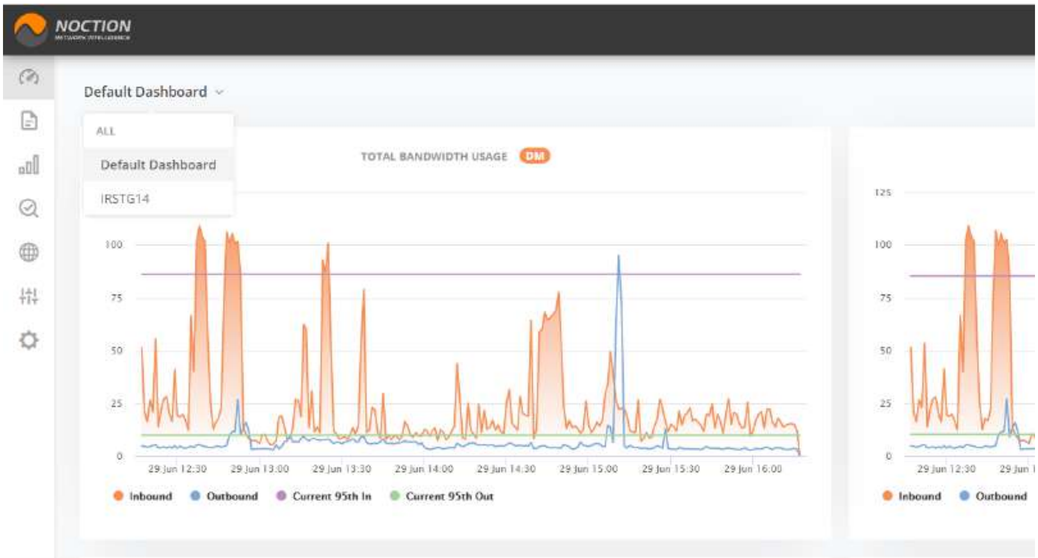

Figure 3.3.1: Dashboard Selection

Dashboards are grouped for easy access into favorite and all. For each dashboard, the directory displays the following information:

Name The name of the dashboard

Description Dashboard user-defined description

Favorite a state marked by a star icon

List of widgets the names of widgets used in the dashboard

Default status the default dashboard the user lands on when logging into GMI

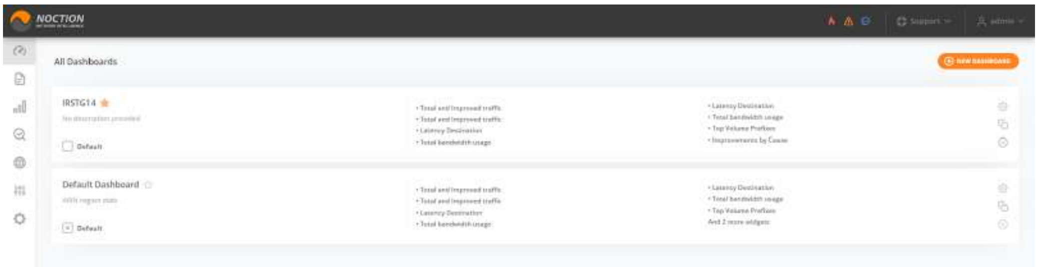

Figure 3.3.2: List of Dashboards

3.3.1 Creating a new dashboard

You can easily create a new dashboard in GMI from the All Dashboards directory.

Click the “NEW DASHBOARD” button at the top right corner of the directory. A pop up will appear. Provide a meaningful name and description for your dashboard.

Mark if you’d prefer it to be a “Favorite” (dashboard will appear at the top of the “All Dashboards” directory) and/or “Default” dashboard.

Press “Create” to continue, or “Close” to return to the directory.

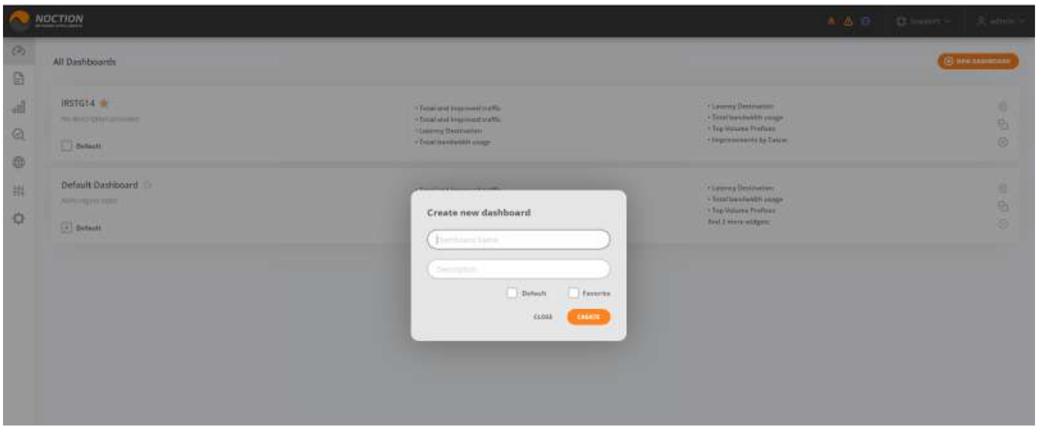

Figure 3.3.3: Creating a new dashboard

Alternatively, you can create a new dashboard by cloning an existing one in the All Dashboards directory. The clone dashboard will be automatically created along with widgets from the original dashboard and added to the directory. Edit the newly created dashboard to change its name and description.

3.3.2 Managing Dashboards

A dashboard can be customized by adding, moving and removing widgets. Hover over any widget to move, edit or delete it. To resize a widget, click the lower right corner and drag it to the desired widget size. Click the display icon in the upper right corner to open a dashboard in full view.

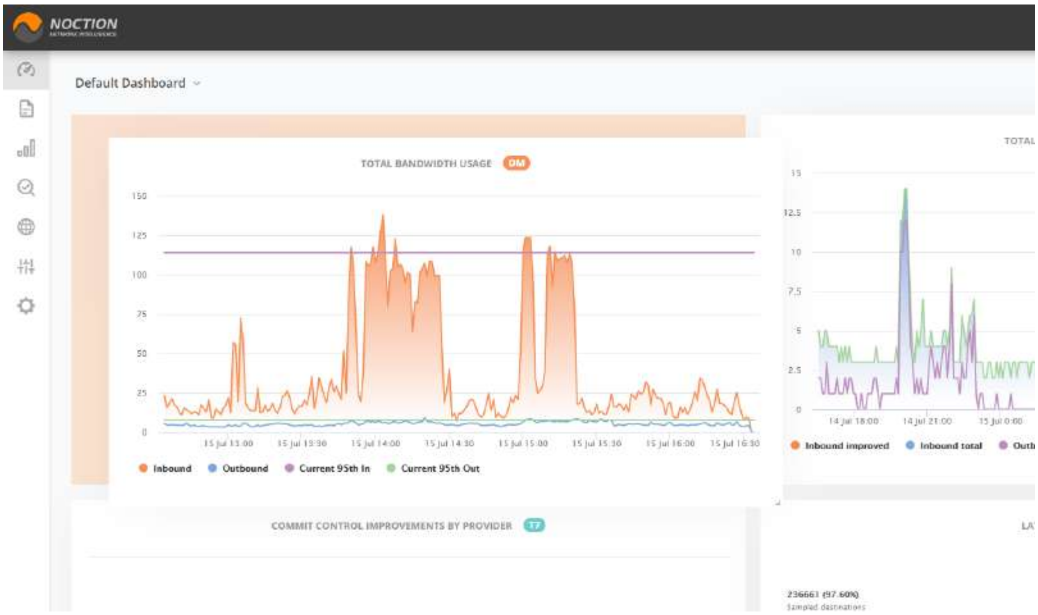

Figure 3.3.4: Rearranging Widgets on Dashboards

Use the "ADD WIDGET" function available on each dashboard to see the library of existing widgets and place the desired ones on a dashboard.

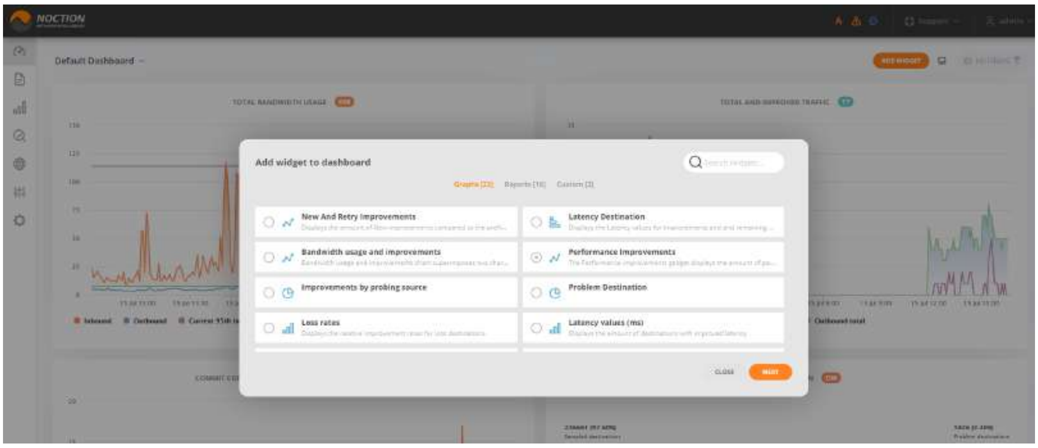

Figure 3.3.5: Widgets Library

You can change the widget's name, refresh interval, timestamp interval as well as the actual IRP instance it is referring to by clicking the Settings icon in the widget's top right corner. Use the rest of the icons to correspondingly refresh, minimize or remove the widget from a dashboard.

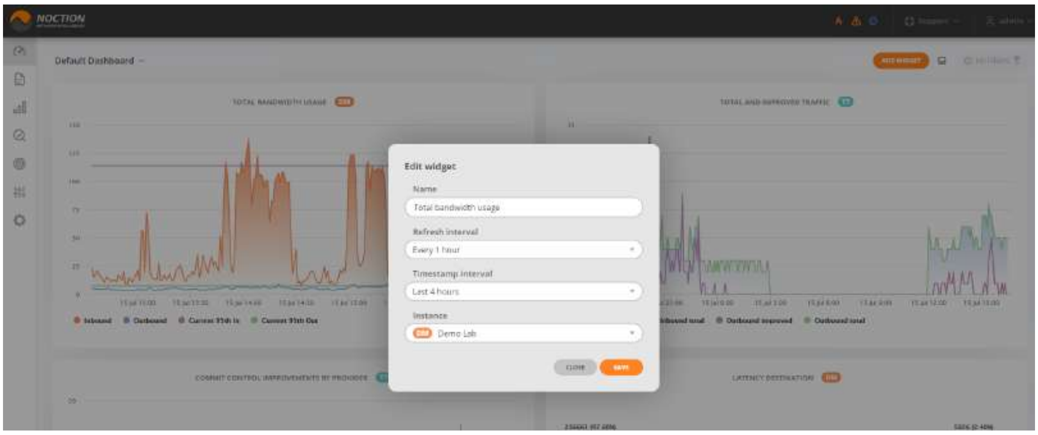

Figure 3.3.6: Editing Widgets

3.3.3 Deleting Dashboards

Click the "Remove" icon on the dashboard you'd like to get rid of in the All Dashboards directory.

Note The default dashboard can not be deleted.

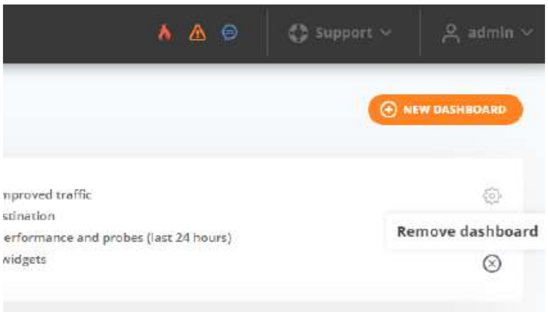

Figure 3.3.7: Deleting Dashboards

3.4 Reports

The GMI Frontend comes with a comprehensive set of reports, reflecting the current state of the network as well as overall statistics on a particular IRP instance performance. Click on the desired report title to open it.

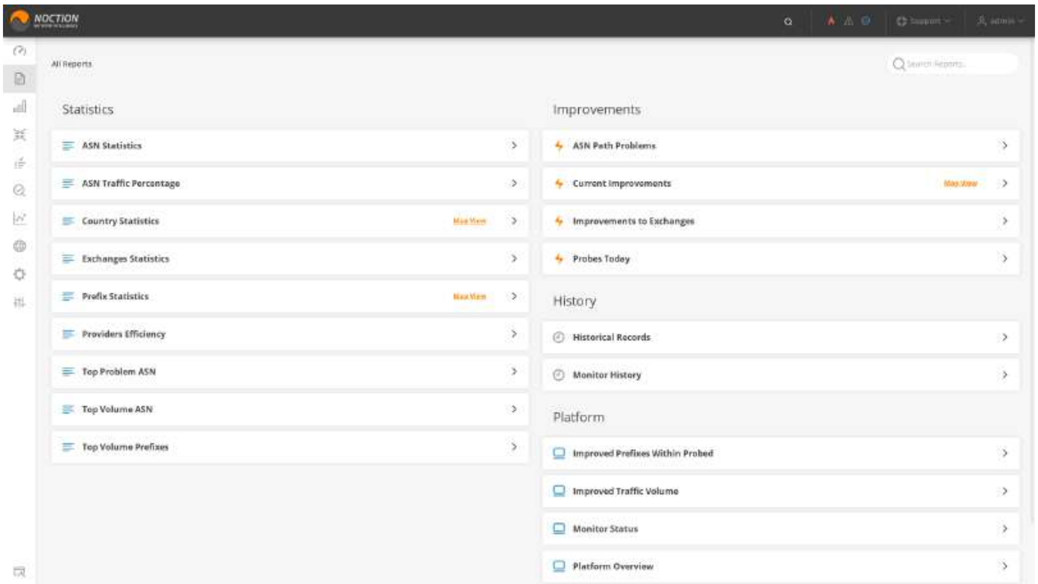

Figure 3.4.1: Reports menu

You can save any report as a widget and add it to any dashboard by clicking the “ADD TO DASHBOARD” button. Select the IRP instance of interest for the report to show relevant data. Filtering options that open up together with a time picker will vary depending on the report type.

Figure 3.4.2: Report options

You can export any report/graph by selecting the desired format (PDF, XLSX, CSV). To print the report, clicking the corresponding print icon.

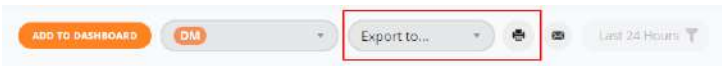

Figure 3.4.3: Report export and print options

Feel free to subscribe to any report by clicking on the “envelope” icon and filling out fields in the popup form.

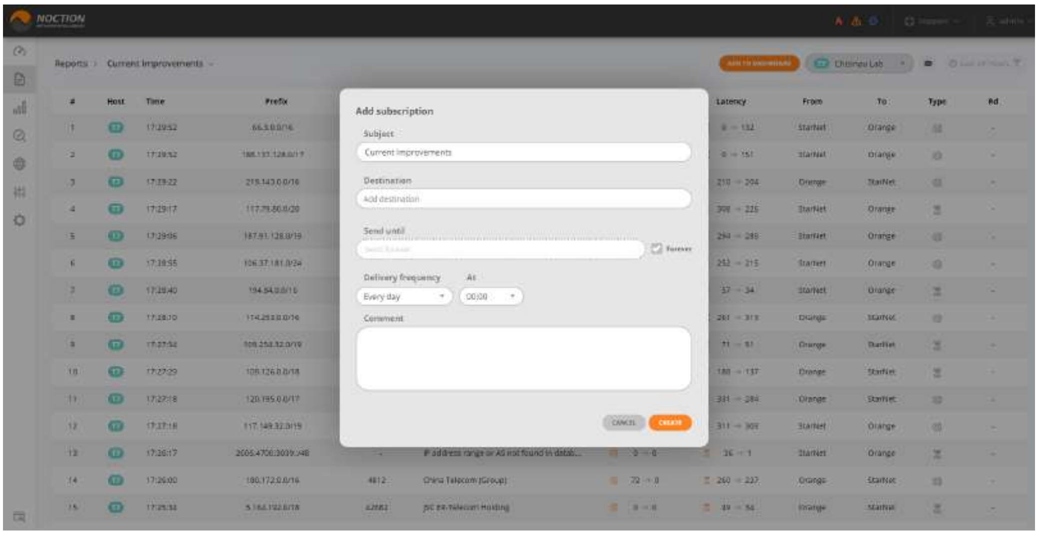

Figure 3.4.4: Report email delivery configuration

When there is a need to compare two separate sets of data within a single report or graph (different filters, time periods, or instances), click the “SAVE FOR LATER COMPARE” button.

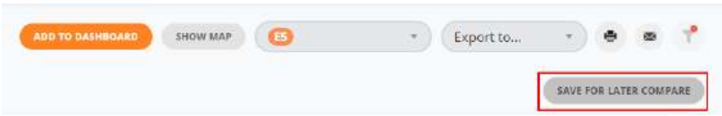

Figure 3.4.5: Save for Later Compare Button

A saved report/graph view will be placed below the current view, allowing you to subsequently introduce new filtering and/or time frame conditions in the top report and compare the two data sets.

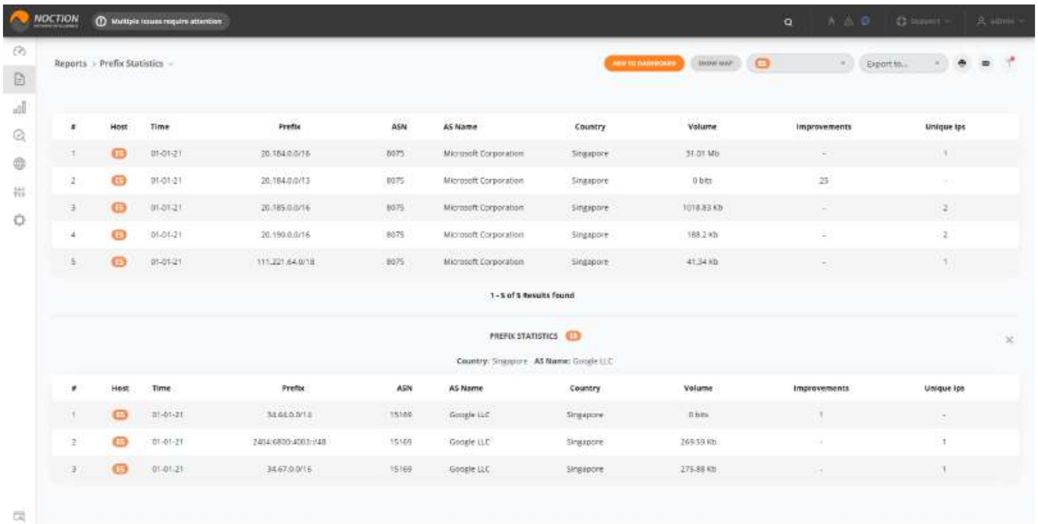

Figure 3.4.6: Save for Later Compare Results

3.4.1 ASN Path Problems

The report is available only when the "Outage detection" is enabled in the IRP Core configuration.

The ASN Path Problems report represents the statistics (congestion and outages) related to the specific AS-PATH sequence, which is selected from IRP probing results and categorized as possible outage and/or congestion.

The report highlights the total number of selected, probed, and pending-for-probing prefixes to either confirm or reject the presence of the congestion or outage for an AS-PATH pattern. When the platform confirms the problem's existence, all of the previously probed prefixes with a similar AS-PATH sequence get rerouted away from the affected path.

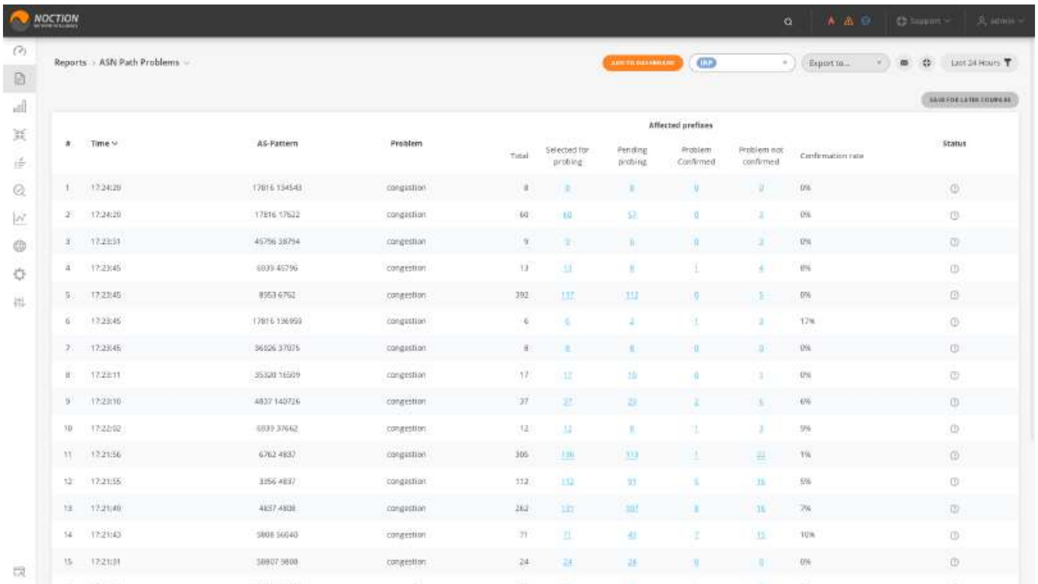

Figure 3.4.7: ASN Path Problems

3.4.2 ASN Statistics

The "ASN Statistics" report aggregates by Autonomous System the total number of improvements as well as the volume of traffic for a selected time period.

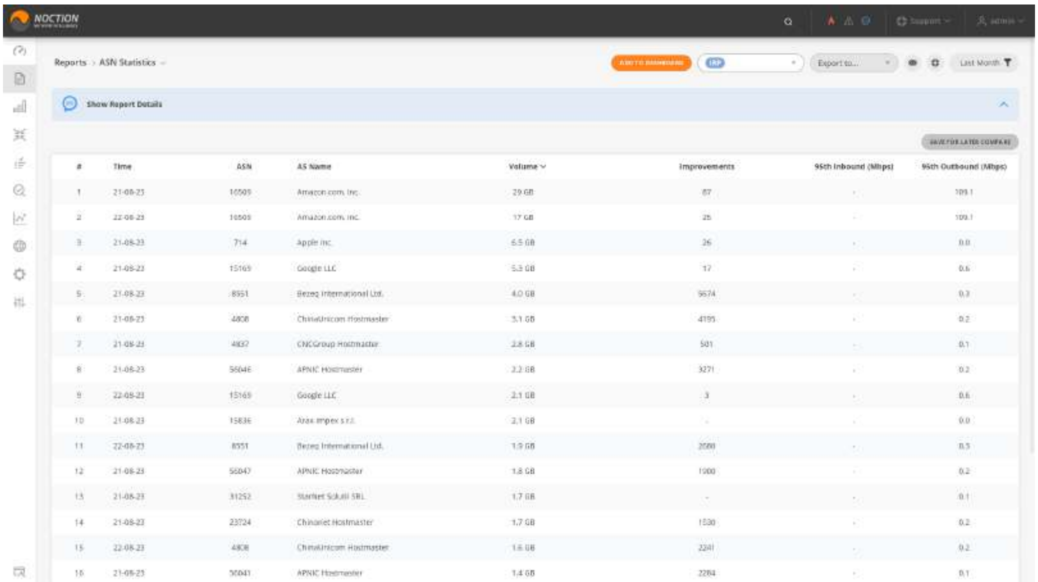

Figure 3.4.8: ASN Statistics

3.4.3 ASN Traffic Percentage

The ASN Traffic Percentage report shows the top sorted Autonomous Systems by the amount of traffic and the calculated percentage of the traffic of these ASes from total traffic.

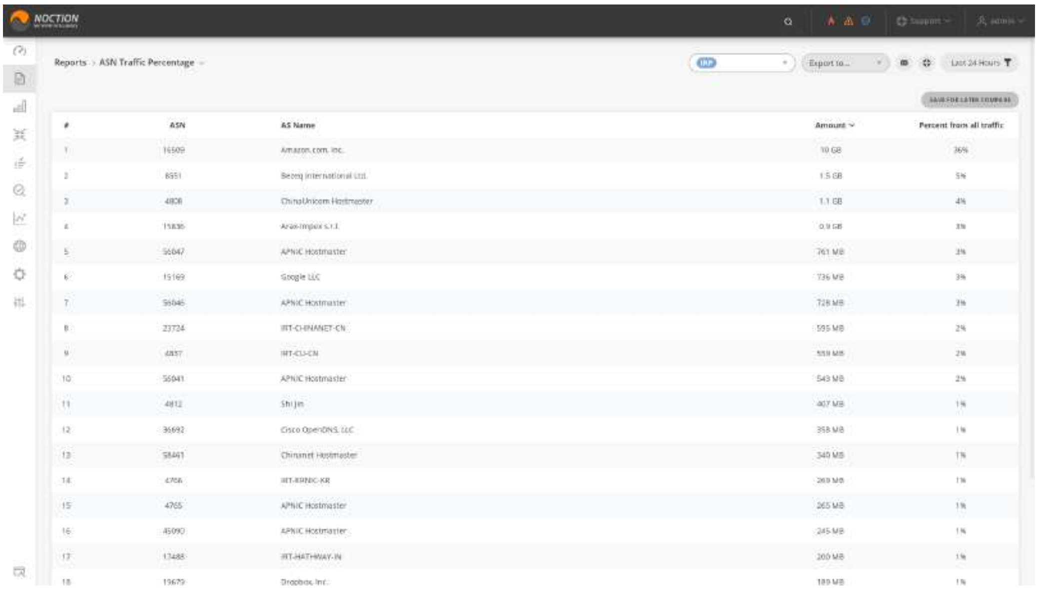

Figure 3.4.9: ASN Traffic Percentage

3.4.4 Country Statistics

The "Country Statistics" report shows the distribution of traffic volume and the number of improvements by country. It helps see to what regions most of a network's traffic is addressed and also the relative number of suboptimal routes towards them that a selected IRP instance identified and addressed by injecting improvements.

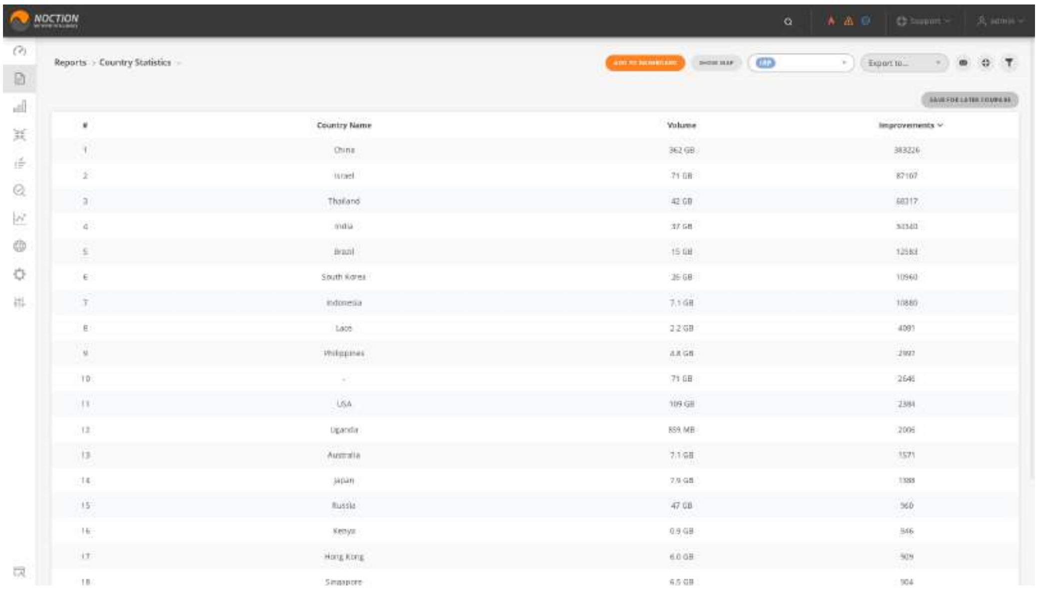

Figure 3.4.10: Country Statistics

Country statistics report offers a map view, highlighting countries with most/least number of improvements.

Figure 3.4.11: Country Statistics map view

3.4.5 Current Improvements

The “Current Improvements” report provides a list of the currently active improvements as per specific IRP instance. The report shows the improved prefix and the providers from and to which the traffic was redirected. It also provides the reason for the improvement, along with the performance values before and after the traffic was rerouted.

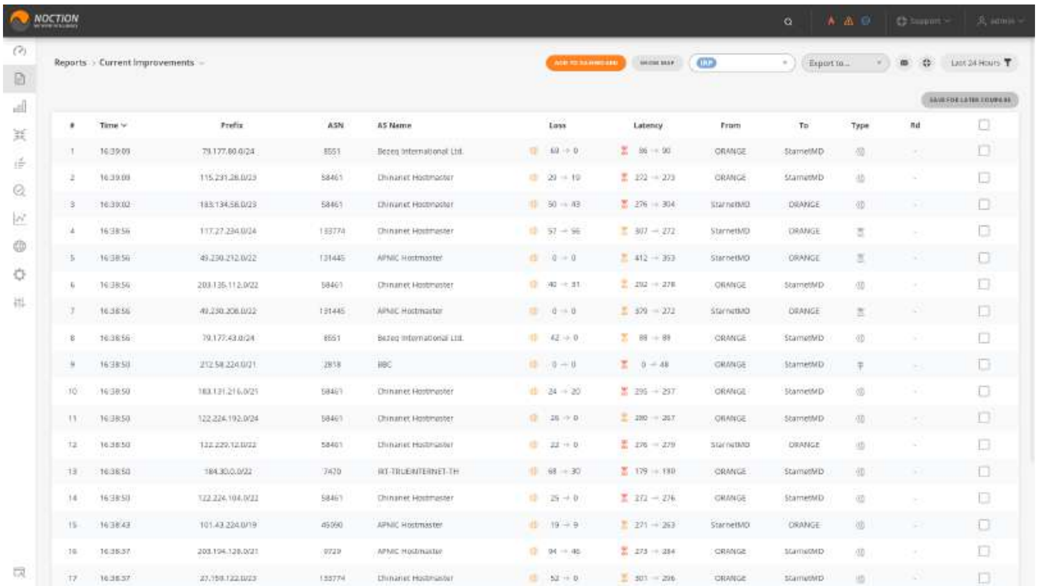

Figure 3.4.12: Current Improvements

> ℹ The “ASN” field can be empty in case there is no information about the AS number (a prefix belongs to) in the ASN dictionary. As soon as the ASN information is available in the dictionary, the field will be filled accordingly.

3.4.6 Exchanges Statistics

The “Exchanges statistics” report lists specific information regarding communication with peers on Internet Exchanges. Filters help identify the required details.

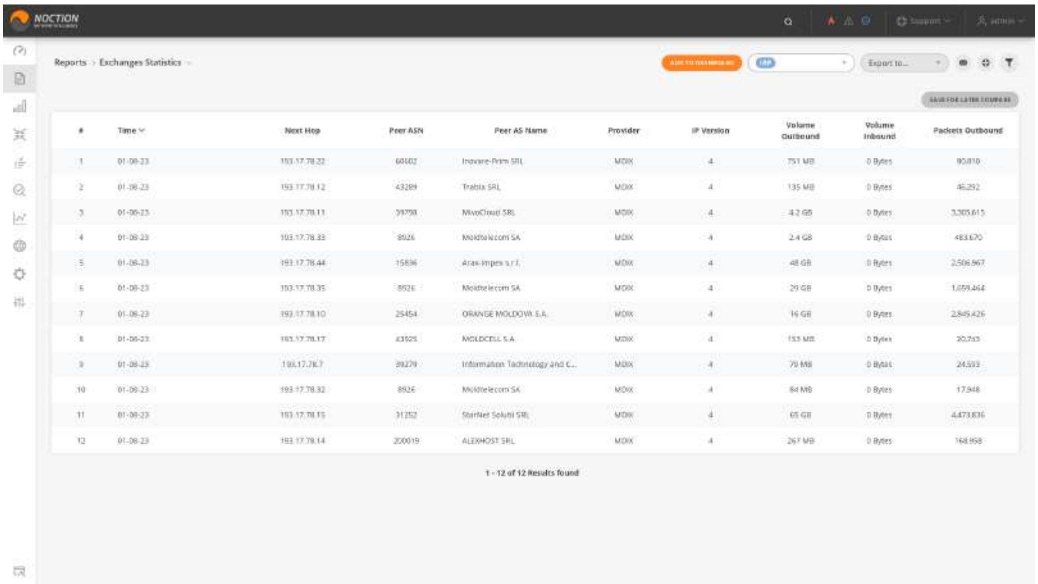

Figure 3.4.13: Exchanges Statistics

3.4.7 Historical Records

The "Historical Records" report contains a complete list of active and inactive improvements that were made by a particular IRP instance.

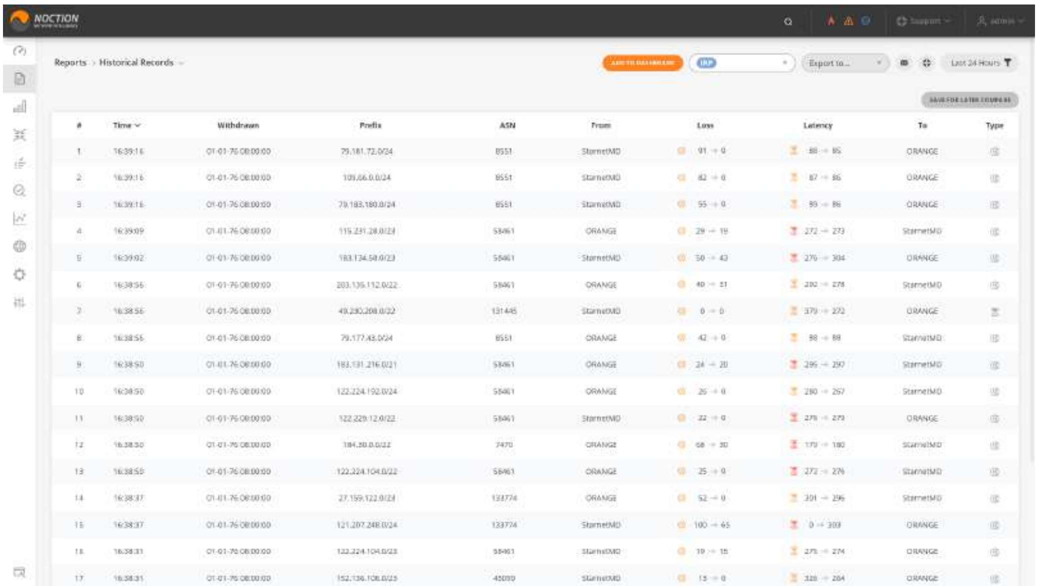

Figure 3.4.14: Historical Records

3.4.8 Improved Prefixes Within Probed

This report offers overall statistics on the number of explored prefixes and the rate of the improved ones per day, week and month.

Figure 3.4.15: Improved Prefixes Within Probed

3.4.9 Improved Traffic Volume

This report offers overall statistics on the amount of total traffic and improved traffic on a daily, weekly and monthly basis.

Figure 3.4.16: Improved Traffic Volume

3.4.10 Improvements to Exchanges

The “Improvements to Exchanges” report contains a complete list of current improvements where the new provider is an Internet Exchange. The report includes data about improved prefixes and their relative traffic by means of last-minute and average bandwidth usage. The average bandwidth is estimated based on current hour statistics.

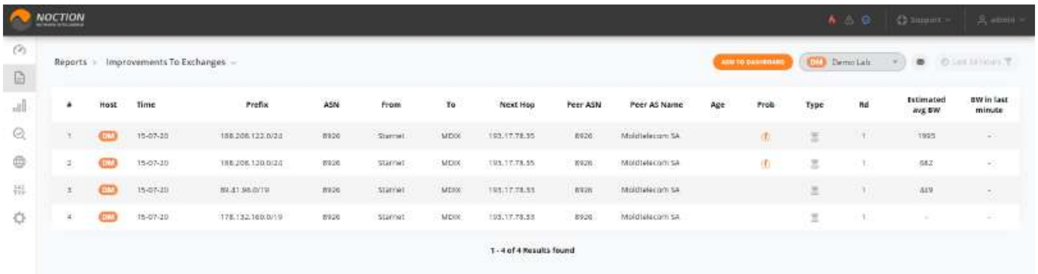

Figure 3.4.17: Improvements to Exchanges

3.4.11 Instances Status (available as a widget only)

The “Instances Status” offers a quick view of the state of IRP instances and their corresponding components.

Figure 3.4.18: Instances Status

3.4.12 Monitor History

The “Monitor History” report highlights the time, monitor details, and the monitor state per provider, facilitating the tracing of various issues on the network and in any particular IRP instance configuration.

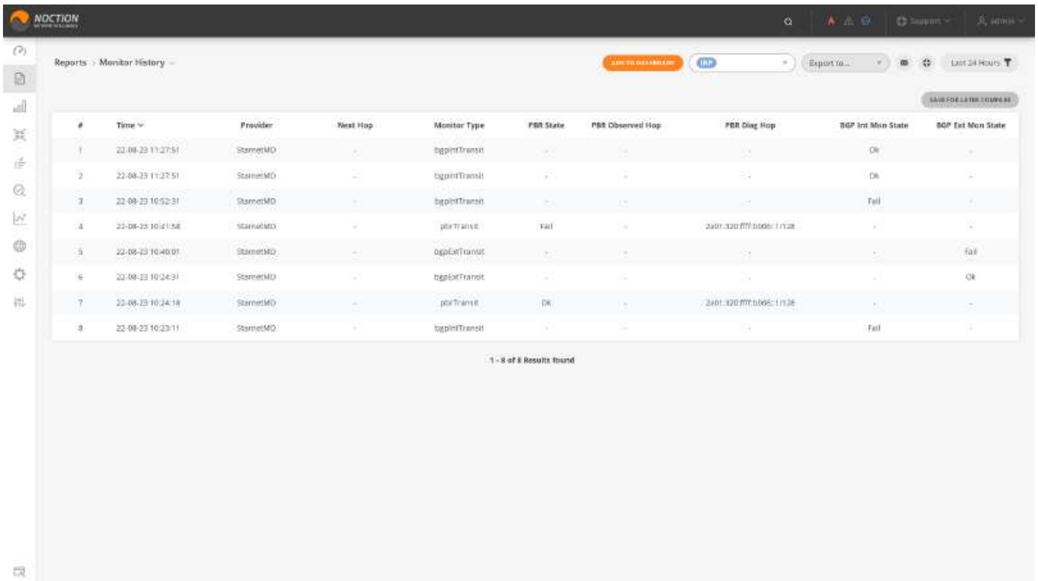

Figure 3.4.19: Monitor History

3.4.13 Monitor Status

The "Monitor Status" will report the status of the IRP BGP Internal and External Monitors for each of the configured providers. BGP Monitors monitor the provider BGP session using SNMP protocol and the ICMP/UDP control packets.

When one of the monitors is reported as failed, IRP will withdraw the improvements made towards this peer.

PBR states as a Policy Based Routing, is the mechanism used to ensure that IRP probing packets follow the preconfigured peer and therefore are correct.

Figure 3.4.20: Monitor Status

3.4.14 Performance Improvements per ASN

Performance Improvements per ASN analysis focuses on the comparison of network performance metrics, specifically packet loss and latency, across different Autonomous System Numbers (ASNs) before and after certain improvements were implemented. The selected time frame for this analysis allows for a comprehensive understanding of the impact these improvements have had on network performance. ASNs are sorted in descending order based on their traffic volume, that helps prioritize ASNs with the highest traffic for a more targeted performance improvement analysis.

The results are presented in a table or graph format, showcasing:

- ASN Identifier: Unique identifier for each ASN
- ASN Name
- Average Packet Loss (Before and After): Showing changes in packet loss.
- Average Latency (Before and After): Indicating changes in latency.

This detailed analysis allows network administrators and stakeholders to evaluate the effectiveness of recent improvements, identify key areas where further enhancements are necessary, and ultimately ensure a more reliable and efficient network performance.

Note! Average latency after improvements may sometimes be higher than before, as reducing packet loss is always prioritized over lowering latency.

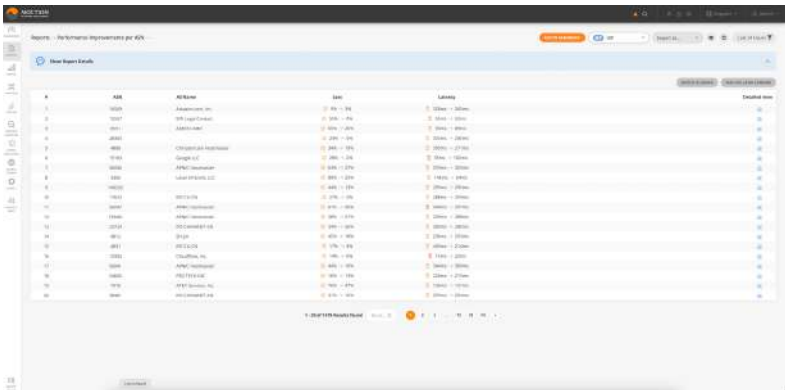

Figure 3.4.21: Performance Improvements per ASN

3.4.15 Platform Overview

The "Platform Overview" provides information on the rate of improvements based upon latency, loss and cost for a particular IRP instance. This report can be used for a quick overview of an IRP instance performance and improvements. The report displays statistics for IPv4/IPv6 separately, while widget added to a dashboard shows aggregated data.

One can see here that almost all the improved IPv4 routes had a loss drop of 20% or more. For 61% of the improved routes the loss was fully eliminated. There are also 16% of the improved routes, which were redirected from a complete blackout. The report also provides the amount of the accomplished route improvements made, based upon latency, cost and commit level.

Figure 3.4.22: Platform Overview

3.4.16 Prefix Improvement Rate by Cause

This report offers overall statistics on improvements rate, based upon performance reasons, commit control, and cost reasons on a daily, weekly, and monthly basis.

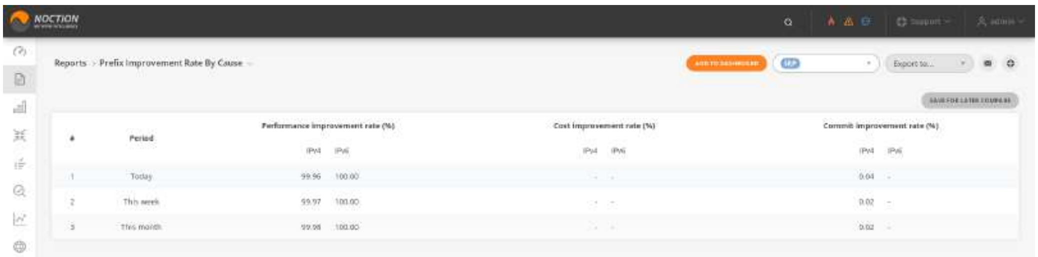

Figure 3.4.23: Prefix Improvement Rate by Cause

3.4.17 Prefix Statistics

This report lists specific prefixes with relevant monthly values such as traffic volume and number of improvements or current unique and top hosts values for a selected IRP instance.

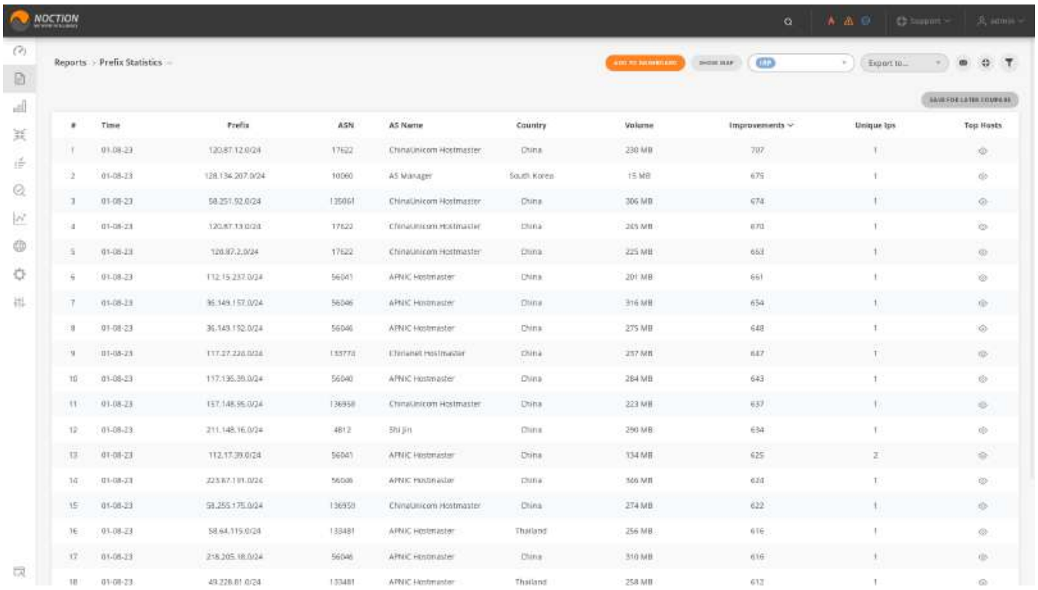

Figure 3.4.24: Prefix Statistics

Note that Prefix Statistics report offers a map view with the regions navigation options, highlighting the parts of the world where either most traffic or the highest number of improvements were made.

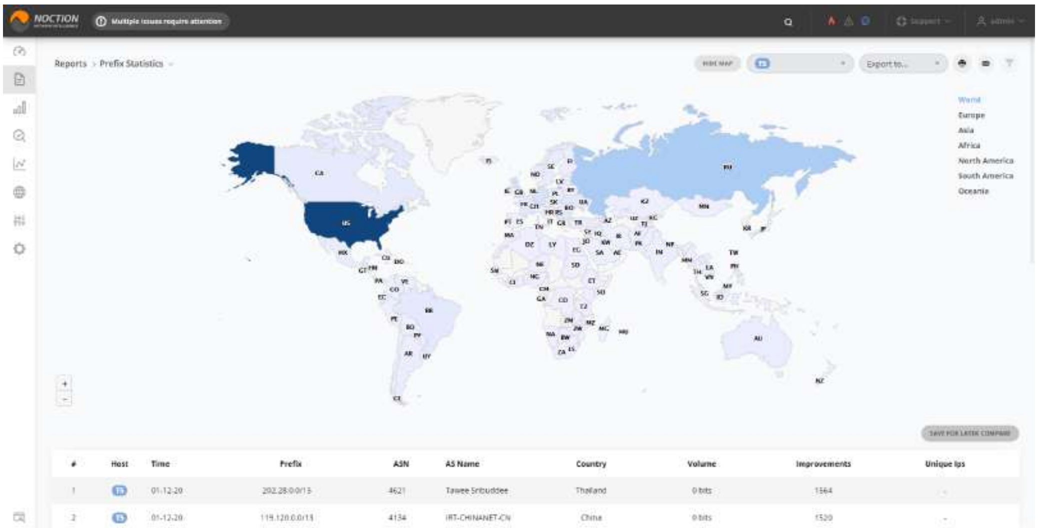

Figure 3.4.25: Prefix Statistics map view

3.4.18 Probes Today

The "Probes Today" report provides probing details, including probes that did not warrant an improvement or failed completely. The report shows probed prefixes and the selected probing IPs in that prefix with details about probe state and actual measurements across all providers. Missing details for some providers, for example for partial providers or Internet exchanges, indicate that either the provider was stopped, the prefix is not serviced by the provider, or a probing IP was not identified and the probe has failed completely.

Icons and coloring are used in the report to highlight data for example if an improvement exists for the prefix, or whether there's a 100% loss. Filtering can be used to quickly analyze the provided data.

See also: Probes Today.

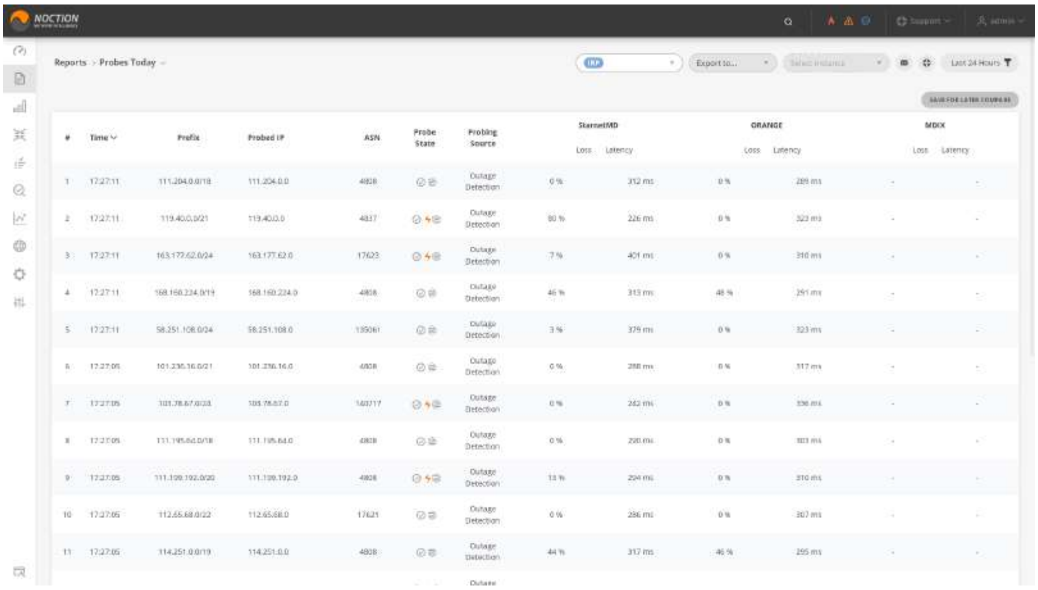

Figure 3.4.26: Probes Today

3.4.19 Providers Efficiency

When operating in any given network, IRP makes numerous probes to determine the performance characteristics of alternative routes.

Provider performance report presents an aggregated view of these measurements. The data on the report highlights the average Packet Loss and average Latency as well as the number of successful and failed probes performed by the platform per each provider.

See also: Providers Efficiency

Figure 3.4.27: Providers Efficiency

3.4.20 Top Problem ASN

The "Top Problem ASN" report reveals the most problematic Autonomous Systems that the monitored networks are sending traffic to. It does that by showing how many times the traffic going to that AS was rerouted by a particular IRP instance.

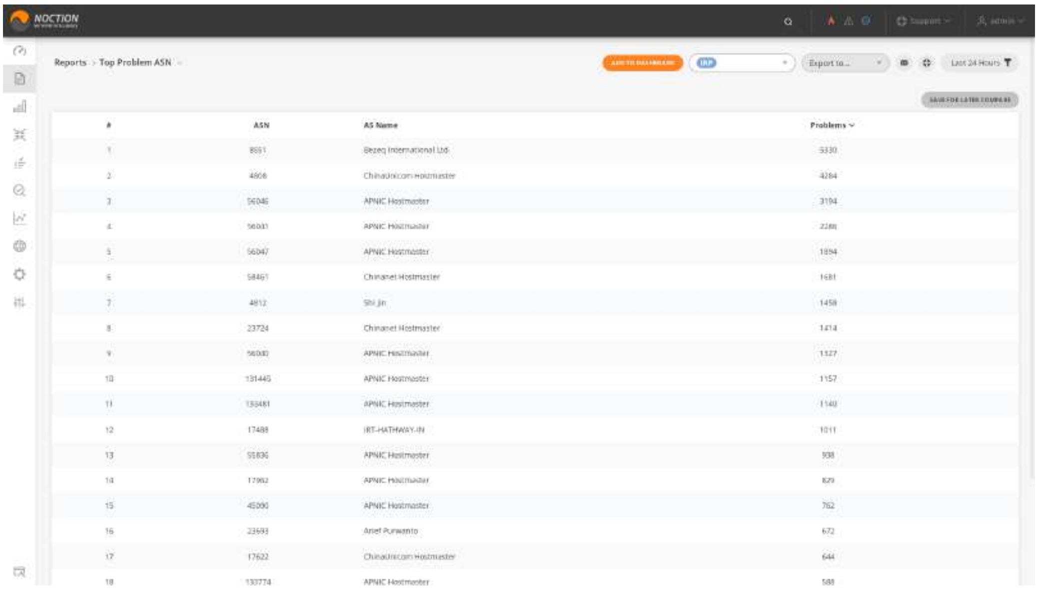

Figure 3.4.28: Top Problem ASN

3.4.21 Top Volume ASN

The "Top Volume ASN" report shows the Autonomous Systems with the highest volume of traffic coming from the monitored network. It can help you understand your traffic flow directions. If according to this report a specific AS has significantly more traffic than the others, then the network operator may consider getting a direct connection to this AS, to reduce traffic costs.

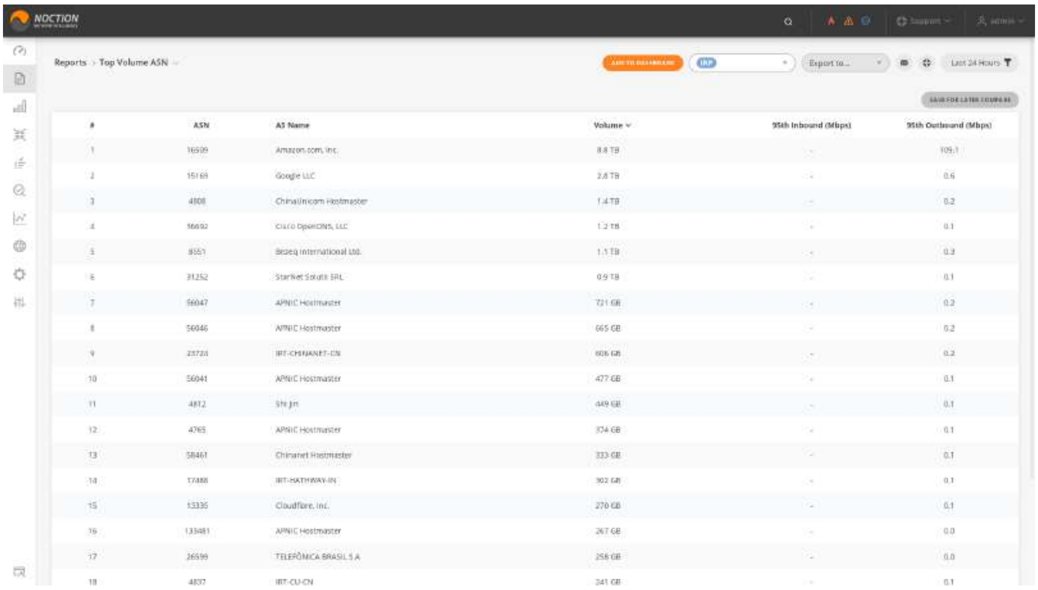

Figure 3.4.29: Top Volume ASN

3.4.22 Top Volume Prefixes

The "Top volume prefixes" displays a list of prefixes sorted by the total volume transferred to these remote networks.

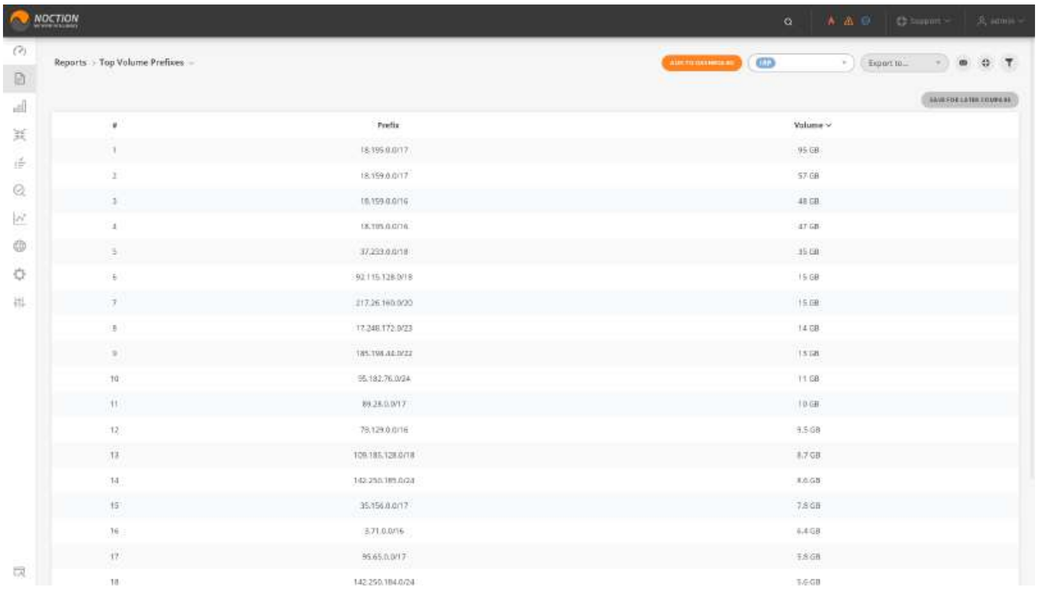

Figure 3.4.30: Top Volume Prefixes

3.5 Graphs

The GMI graphs are visual representations of data found in the reports. They provide an opportunity to quickly identify problems in the network, as well as forecast periodic changes in the traffic patterns of specific IRP instances and the overall network behavior. For each graph you can adjust the time period which you want to get the results for.

> Unless otherwise noted, all graphs are plotted using 5 minute intervals.

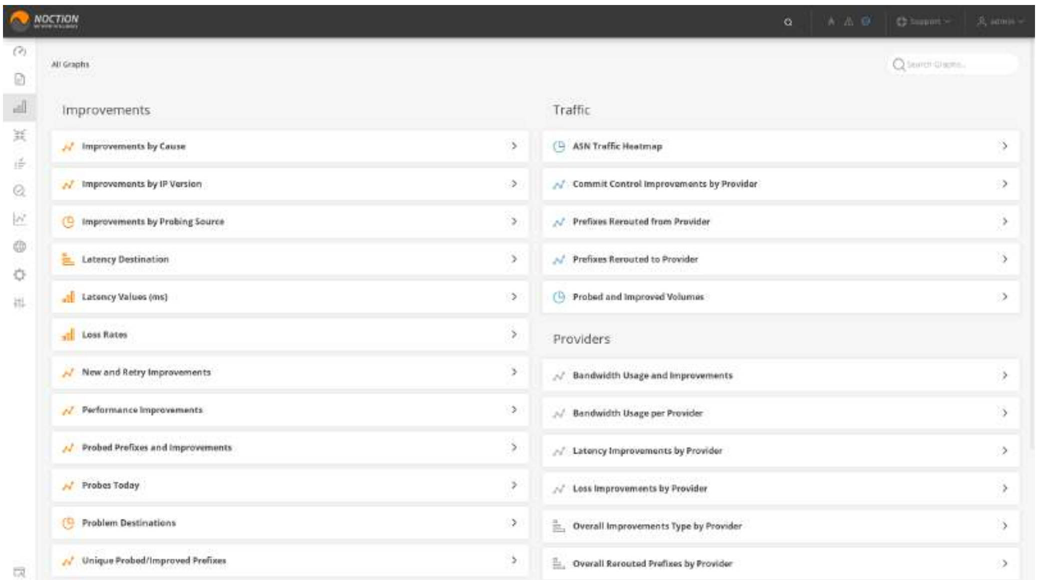

Figure 3.5.1: Graphs menu

To add any given graph to a dashboard, export it or print, consult the instructions provided in section 3.4

3.5.1 ASN Traffic Heatmap

ASN Traffic Heatmap is the graphical illustration of the AS traffic percentage which highlights the total amount of traffic by ASN and the calculated percentage from total traffic.

Each ASN is represented by a colored heatmap cell, with darker shades indicating higher traffic volume and lighter shades indicating lower traffic volume. The graph legend explains the color coding scheme used in the heatmap further.

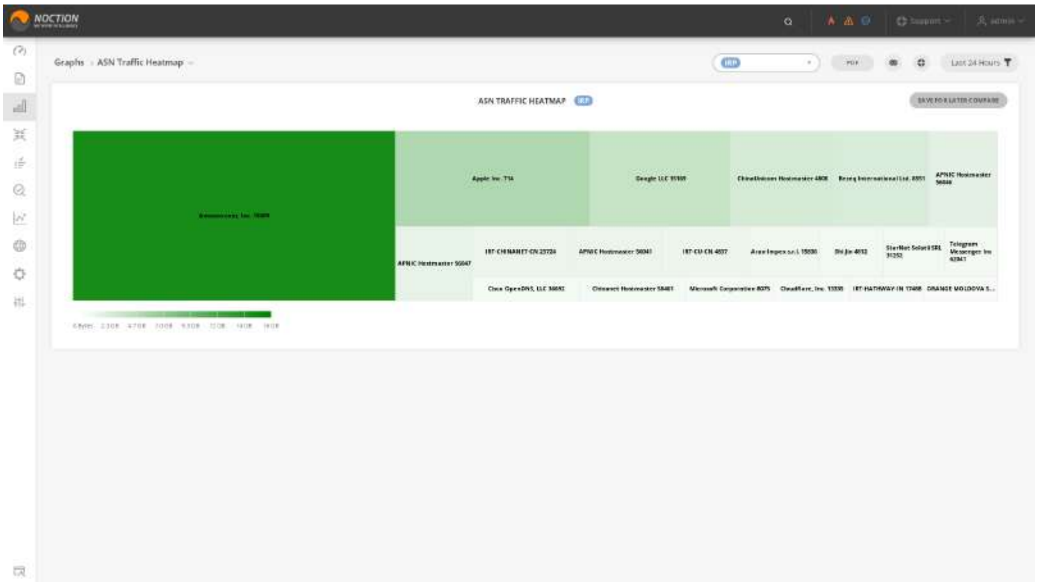

Figure 3.5.2: ASN Traffic Heatmap

3.5.2 Bandwidth Usage and Improvements

Bandwidth usage and improvements chart superimposes graphs for easy analysis and comparison of how improvement counts correlate with bandwidth.

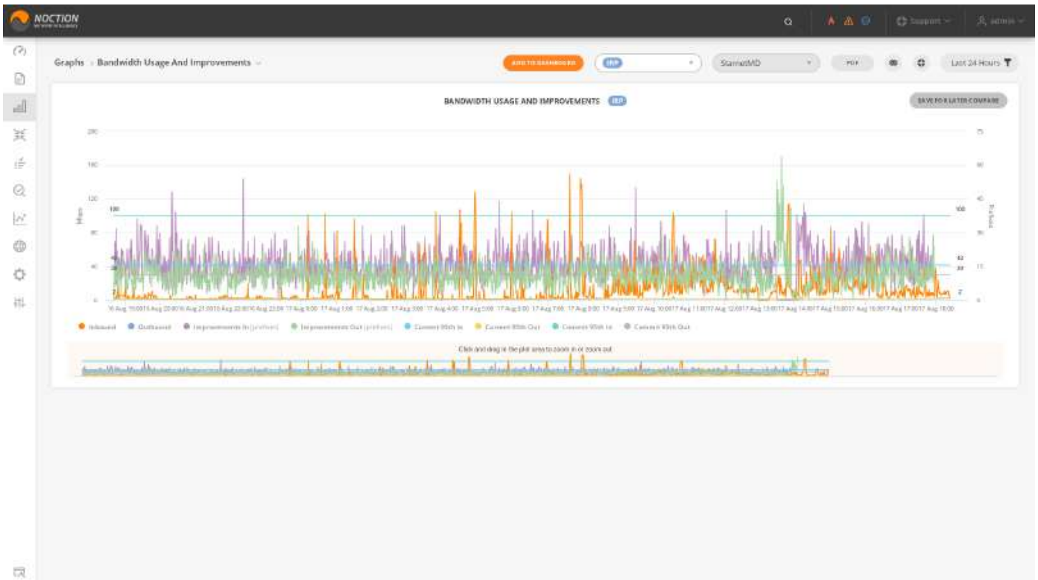

Figure 3.5.3: Bandwidth Usage and Improvements

3.5.3 Bandwidth Usage per Provider

The Bandwidth Usage graph shows the total usage for each of the providers. It allows you to compare the current traffic volume to the current 95th percentile and the commit 95th. The graph provides average, maximum and minimum traffic usage values for each of the providers during the selected time period.

Figure 3.5.4: Bandwidth Usage per Provider

3.5.4 Commit Control Improvements by Provider

If the Commit Control algorithm is enabled on a selected IRP instance, this graph will display an overview of all the commit improvements that were enforced by the platform for a specific time period.

Figure 3.5.5: Commit Control Improvements by Provider

3.5.5 Improvements by Cause

The "Improvements by Cause" graph shows the improvements made by a specific IRP instance based upon performance, cost, commit, and outage detection. Various problem patterns can be detected in compliance with the platform's activity shown in this graph. Different patterns can be noticed depending on the improvement mode that is currently running.

Figure 3.5.6: Improvements by Cause

3.5.6 Improvements by IP Version

This graph shows the number of improved prefixes by either IPv4 or IPv6 protocol.

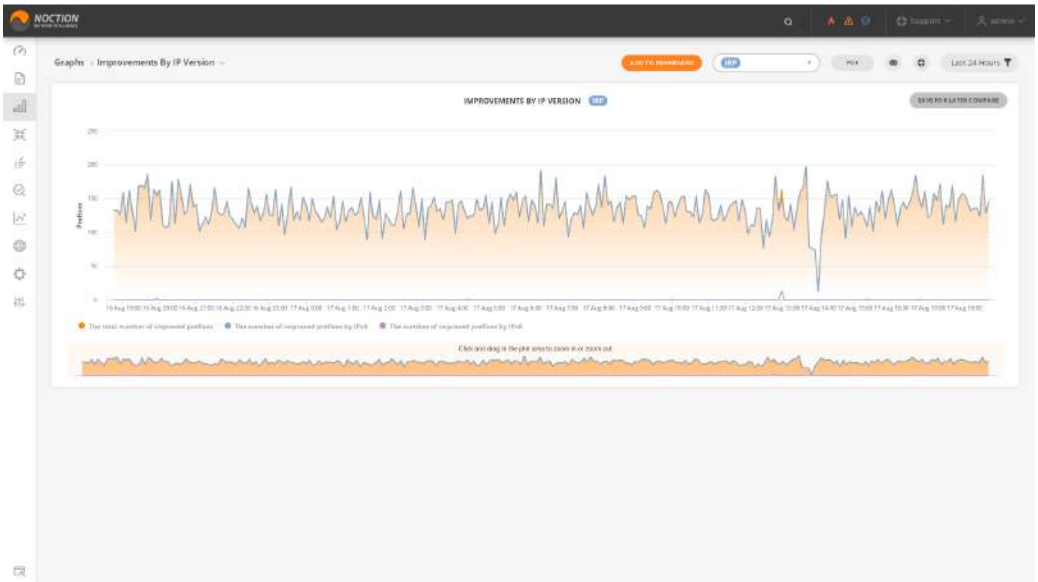

Figure 3.5.7: Improvements by IP Version

3.5.7 Improvements by Probing Source

The graph displays the percentage of improvements that are differentiated by the probing source including Commit Control, Outage Detection, VIP Probing, Regular, and Retry Probing.

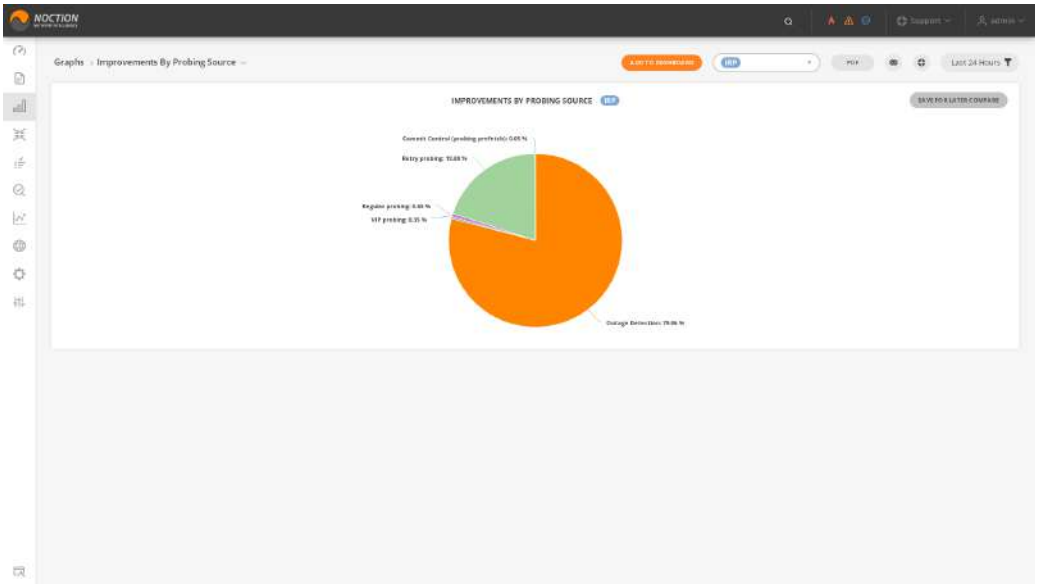

Figure 3.5.8: Improvements by Probing Source

3.5.8 Latency Destination

The Latency Improvements report presents an overview for average latency values before and after specific IRP instance optimization during a selected time period and depicts them across the following populations:

- Sample destinations that cover good and problematic destinations that an IRP instance probed during the selected time period
- Problematic destinations cover those routes/prefixes that IRP assessed to have excessive latency rates and identified a better route via a different upstream provider
- 50% or more cover those destinations where IRP was able to identify a route with latency improvements of 50% or more
- 20% or more cover those destinations where IRP was able to identify a route with latency improvements of 20% or more

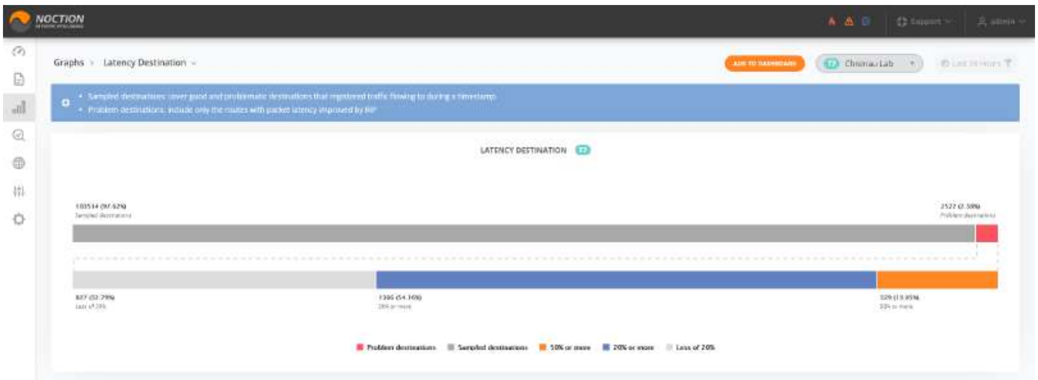

Figure 3.5.9: Latency Destination

3.5.9 Latency Improvements by Provider

Graph displays a representation of the average number of latency-based improvements for each provider per selected IRP instance in five minute intervals. Specific provider lines can be removed from the graph by clicking the provider name in the graph's legend.

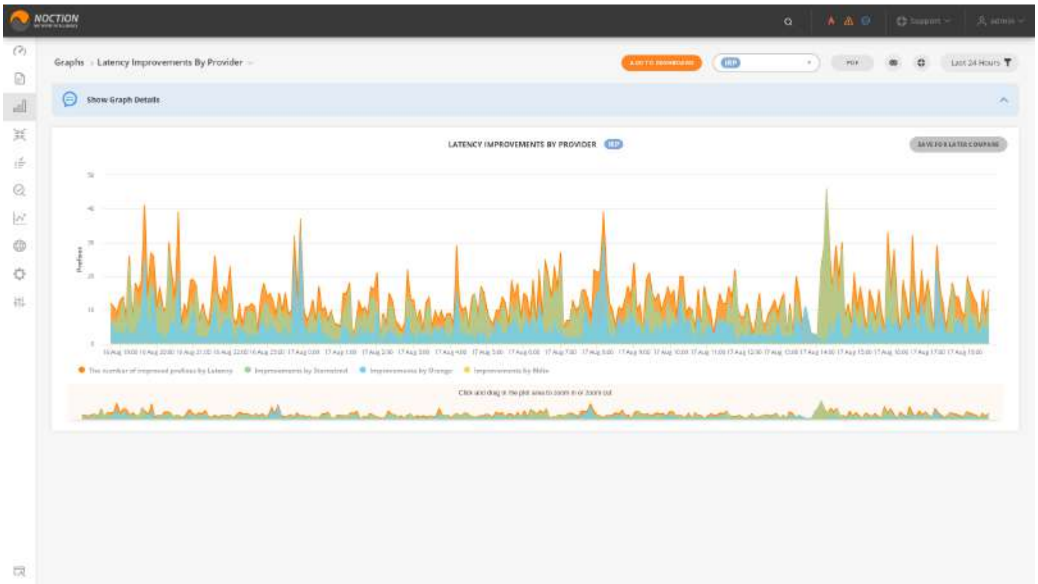

Figure 3.5.10: Latency Improvements by Provider

3.5.10 Latency Values (ms)

Latency values bar-chart highlights the latency averages before and after improvement for the designated groups of destinations.

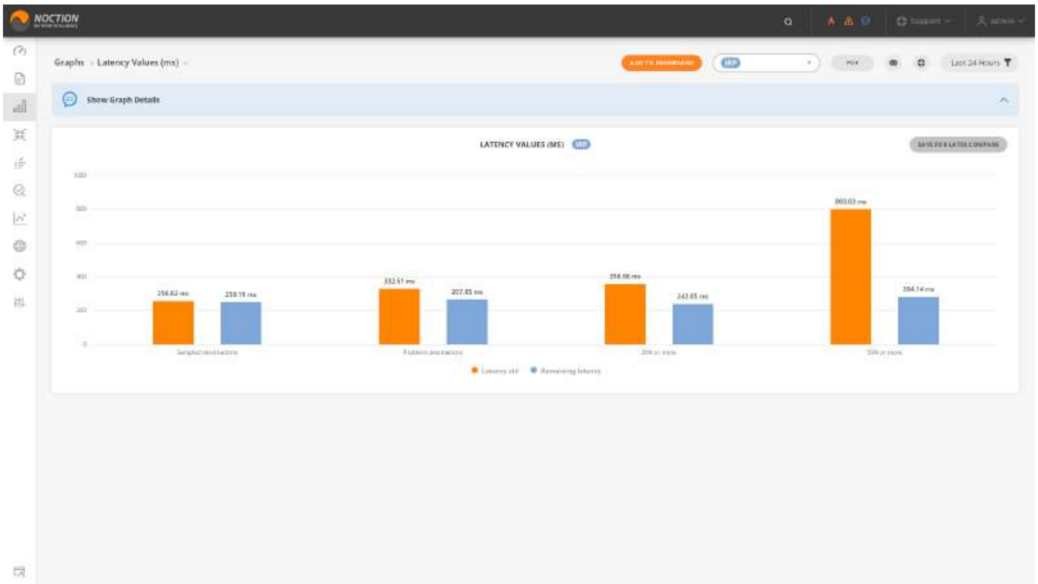

Figure 3.5.11: Latency Values (ms)

3.5.11 Loss Improvements by Provider

Graph displays a representation of the average number of loss-based improvements for each provider per selected IRP instance in five minute intervals.

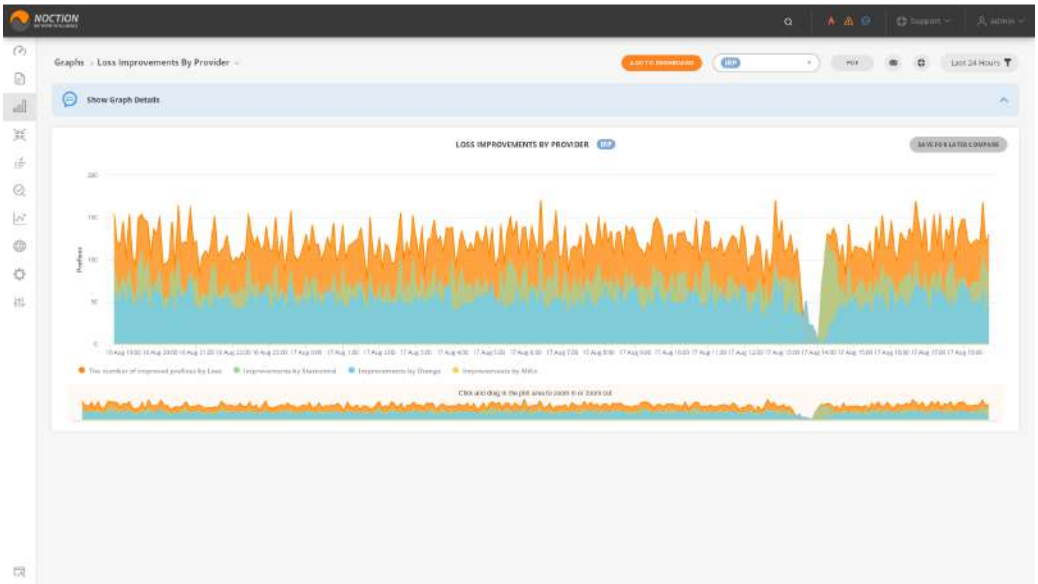

Figure 3.5.12: Loss Improvements by Provider

3.5.12 Loss Rates

Loss values bar-chart highlights the packet loss averages before and after improvement for the designated groups of destinations.

Sampled destinations cover good and problematic destinations that registered traffic flowing to during a selected timeframe. Problem destinations include only the routes with packet loss improved by specific IRP instances.

Loss eliminated displays Problem destinations that an IRP instance was able to improve to a better route with zero loss.

Loss reduced: displays Problem destinations that IRP was able to improve but could not reduce loss altogether.

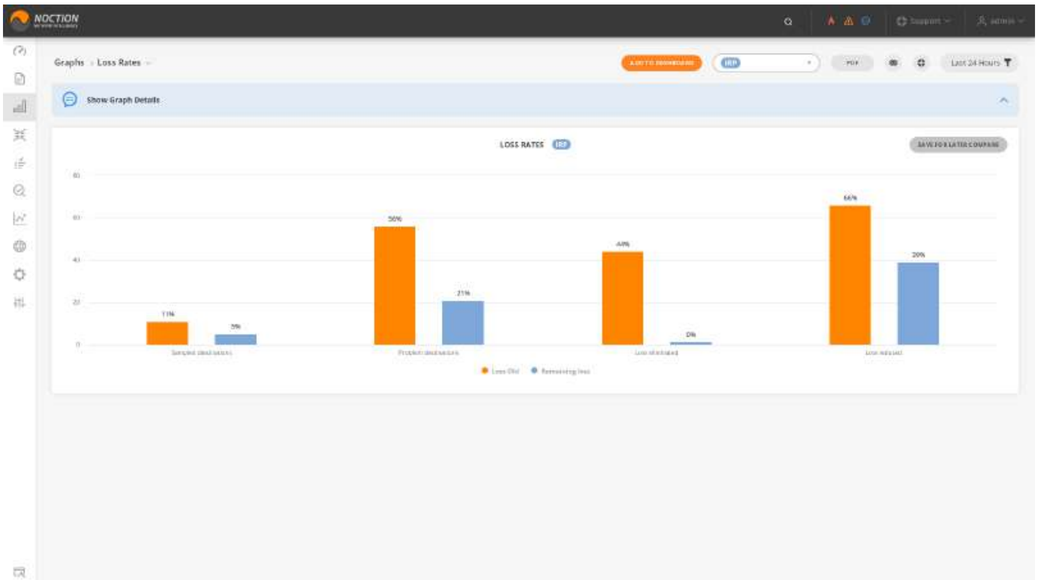

Figure 3.5.13: Loss Rates

3.5.13 New and Retry Improvements

After a particular improvement is made, IRP analyzes the path periodically. If the improvement is still valid, the platform leaves it as it is. The graph shows the number of improvements made by a selected IRP platform for the first time as related to those, which were simply confirmed after re-probing.

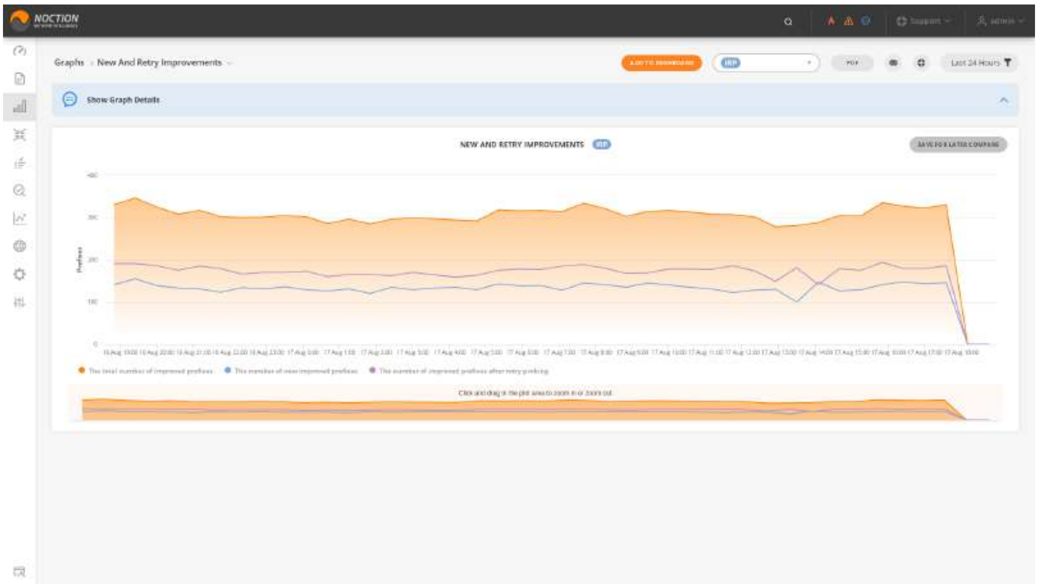

Figure 3.5.14: New and Retry Improvements

3.5.14 Overall Improvements Type by Provider

Graph shows the number of performance improvements vs other types of improvements per provider for a selected IRP instance.

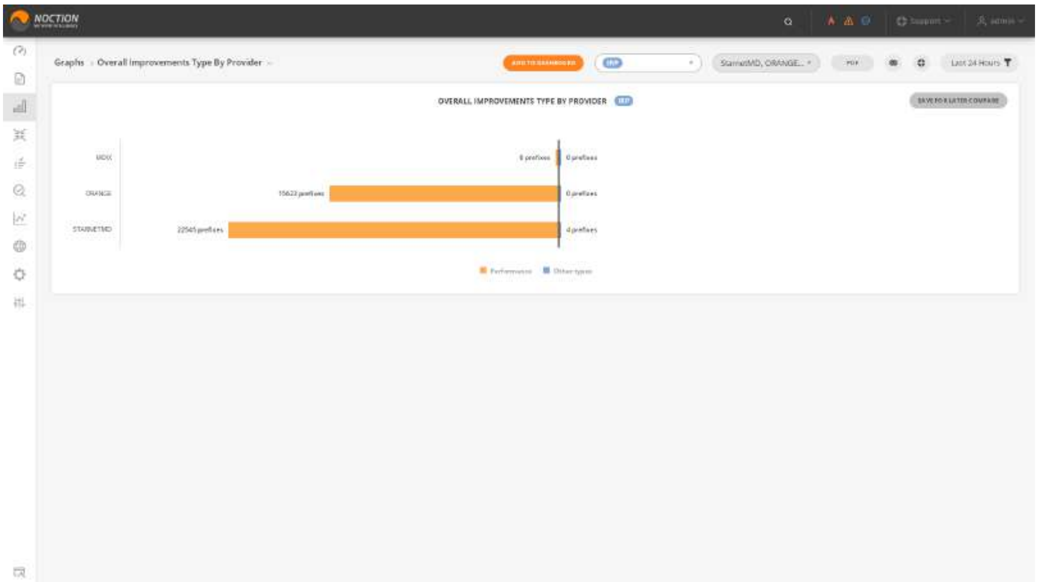

Figure 3.5.15: Overall Improvements Type by Provider

3.5.15 Overall Rerouted Prefixes by Provider

The “Overall rerouted prefixes by provider” graph provides the number of prefixes that were rerouted from and to each of the providers. In the example below, the traffic exchange between the providers is balanced. However, if the difference between the outgoing and incoming traffic is significant, then the quality of that provider's network should be questioned. "In" represents prefixes rerouted to a provider, "Out" represents prefixes rerouted from a provider. The "Excess In/Out" values represent the difference between the number of prefixes rerouted to a particular provider and the number of prefixes rerouted from this provider.

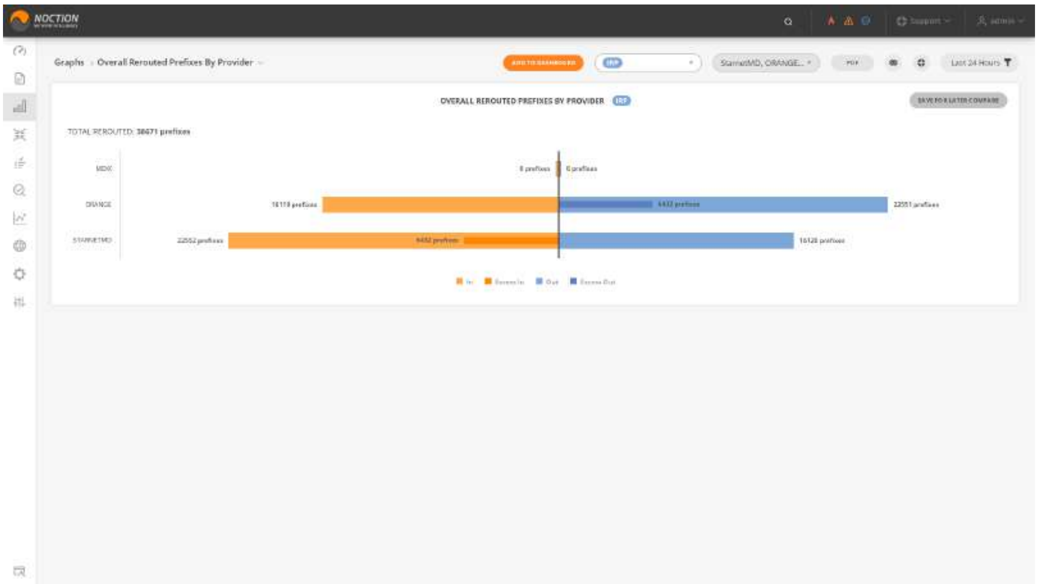

Figure 3.5.16: Overall Rerouted Prefixes by Provider

3.5.16 Performance Improvements

The "Performance Improvements" graph displays prefixes which were improved based upon a performance reason. By following this graph, the solved loss and latency issues, occurring in the network, can be monitored.

This graph provides the average, maximum and minimum number of improvements for each of the improvement reasons during the selected time frame. The maximum peaks are indicators of a loss or latency problem in the network.

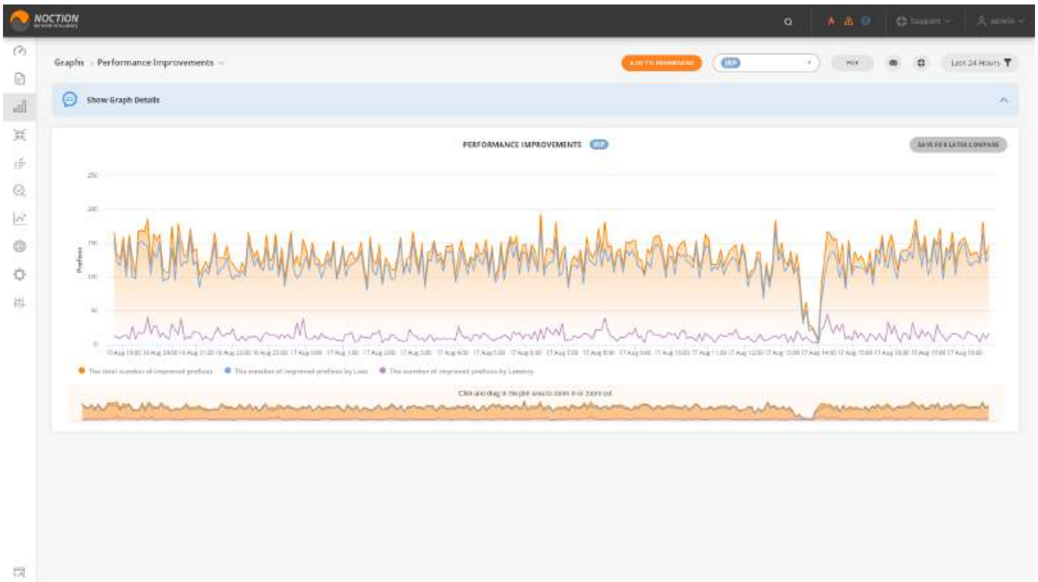

Figure 3.5.17: Performance Improvements

3.5.17 Performance Improvements per Top ASN

The current graph represents averaged loss and latency per top ASN before and after improvement within the selected time frame. Results are sorted in descending order by ASN traffic volume.

Clicking on a specific ASN will redirect you to its dedicated page, displaying daily statistics for the past seven days.

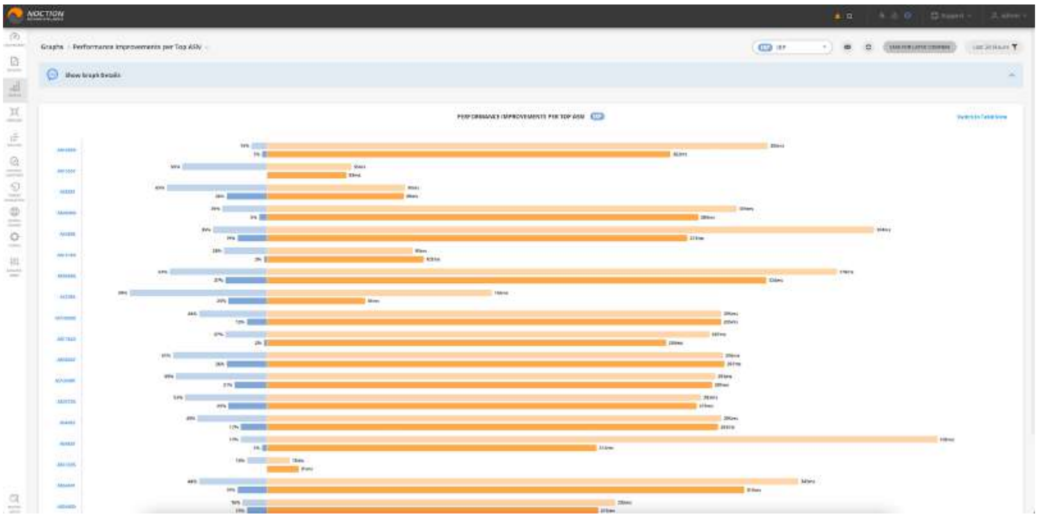

Figure 3.5.18: Performance Improvements per Top ASN

3.5.17.1 Performance Improvements per ASN

The current graph represents daily loss and latency statistics for the past seven days for the selected Autonomous System number.

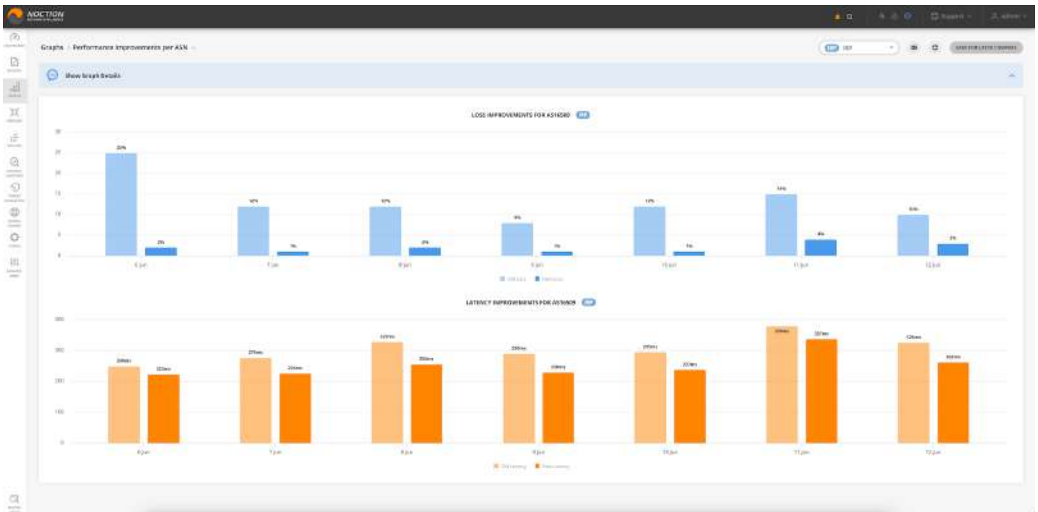

Figure 3.5.19: Performance Improvements per ASN detailed view

3.5.18 Prefixes Rerouted from Provider

The current graph displays the number of prefixes that were rerouted from a specific provider in order to resolve performance or cost issues.

Figure 3.5.20: Prefixes Rerouted from Provider

3.5.19 Prefixes Rerouted to Provider

The graph displays the number of prefixes that were rerouted to a specific provider in order to resolve performance or cost issues.

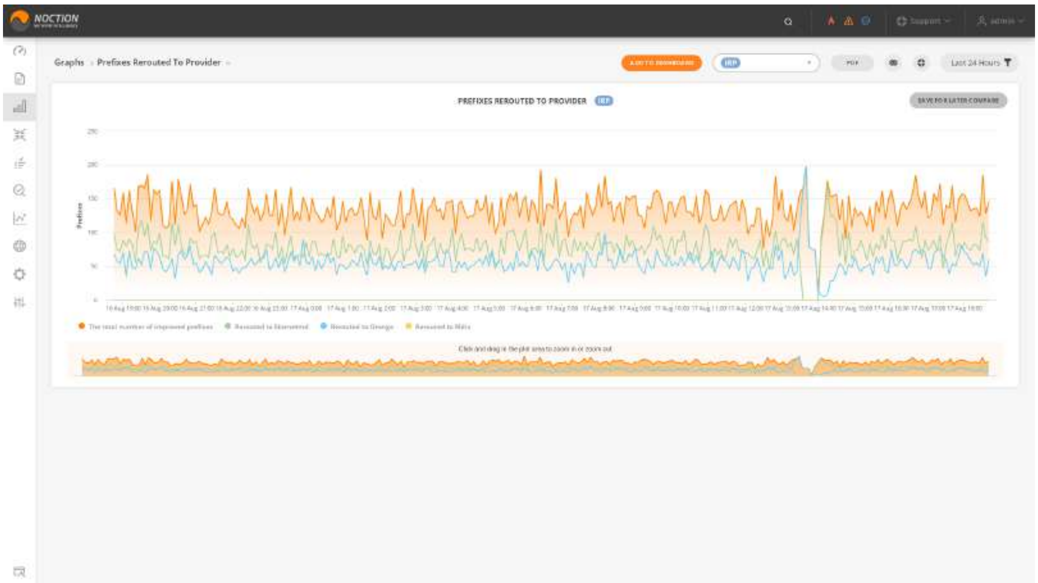

Figure 3.5.21: Prefixes Rerouted to Provider

3.5.20 Probed and Improved Volumes

The pie chart graph displays the amount of traffic volume improved by a particular IRP instance out of the total traffic volume.

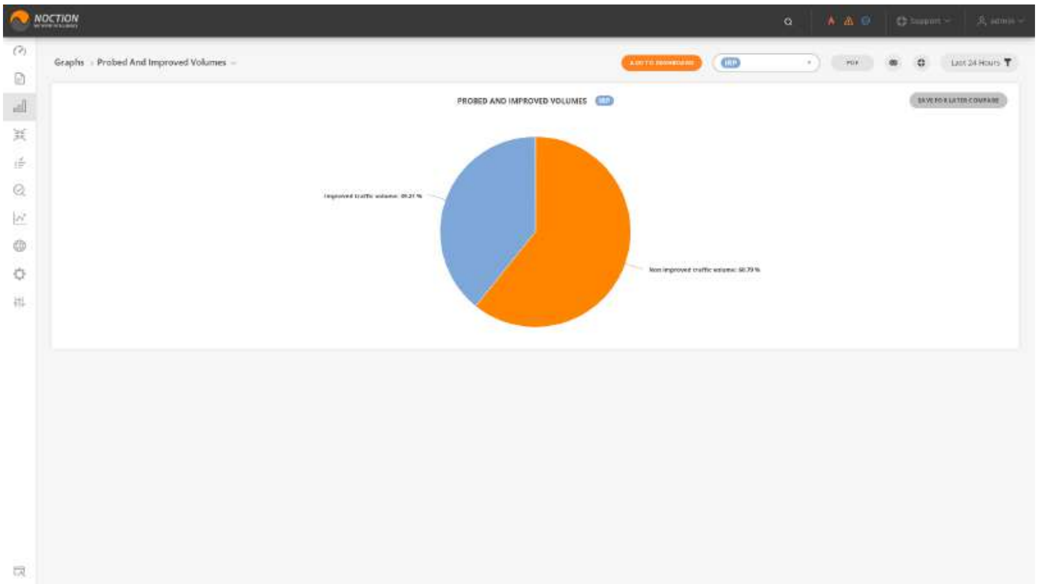

Figure 3.5.22: Probed and Improved Volumes

3.5.21 Probed Prefixes and Improvements

The "Probed prefixes and Improvements" graph shows the number of improvements (Loss, Latency, Commit Control and Other) per selected time period as well as the amount of total probed prefixes.

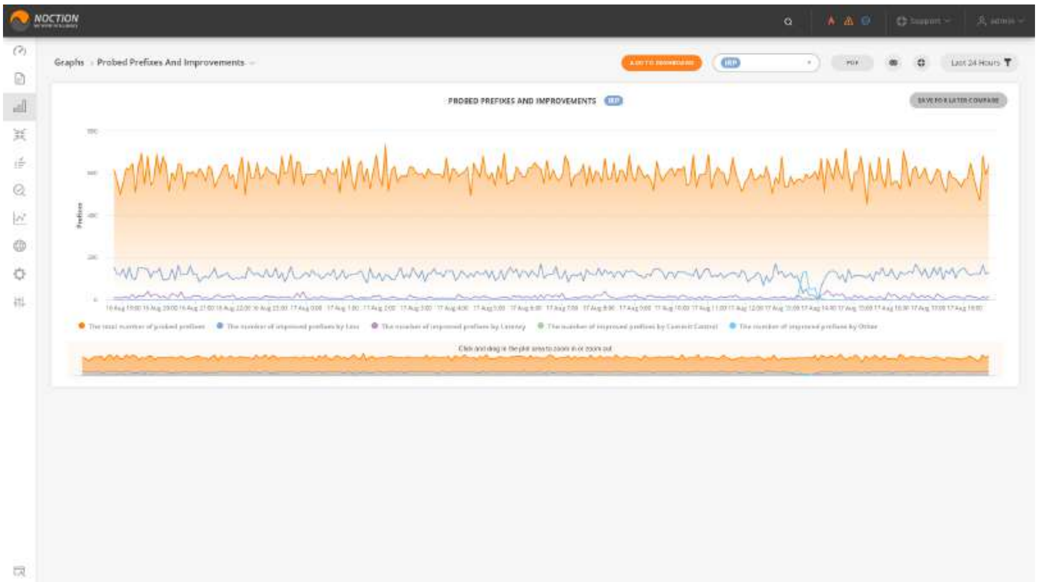

Figure 3.5.23: Probed Prefixes and Improvements

3.5.22 Probes Today

The “Probes Today” graph provides a visual representation of the probes performed within the last 24 hours, including probes that did not warrant an improvement or failed completely.

See also: Probes Today.

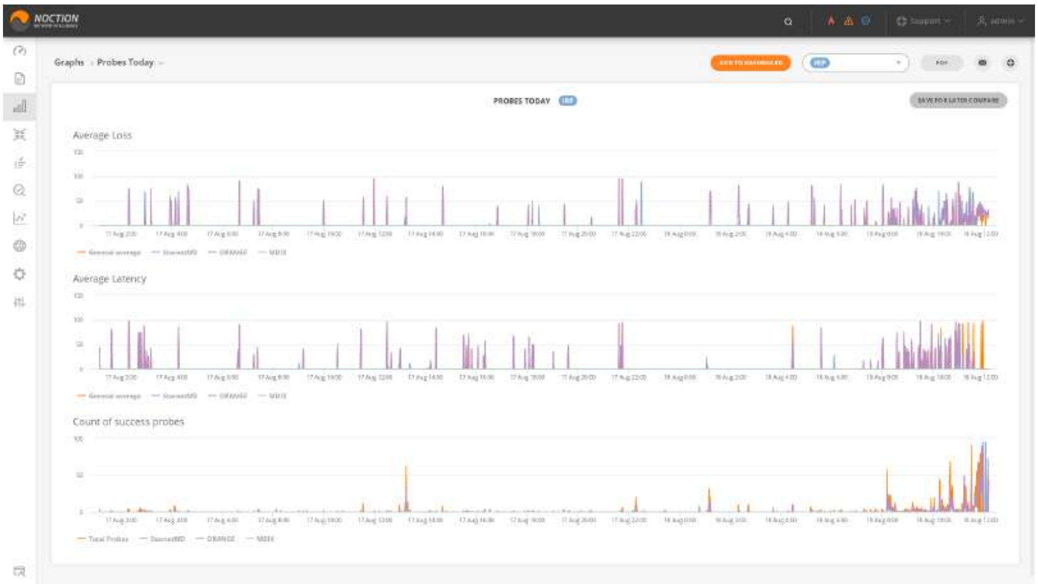

Figure 3.5.24: Probes Today

3.5.23 Problem Destinations

Represents a pie chart with the distribution of improved destinations based on Packet Loss reason. It shows for what percentage of destinations Packet Loss was reduced and for what percentage of problematic destinations Loss was eliminated completely.

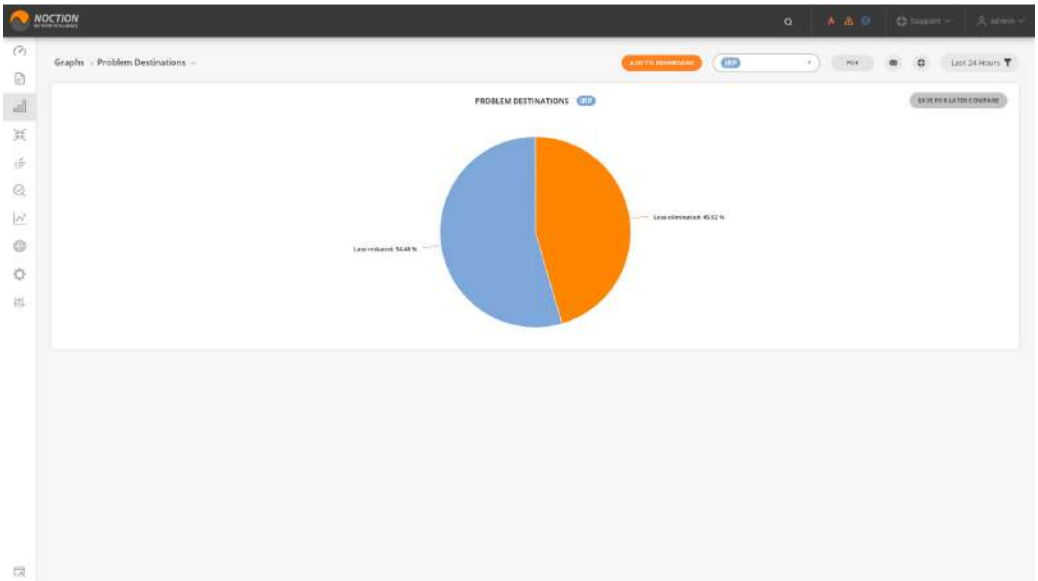

Figure 3.5.25: Problem Destinations

3.5.24 Providers Bandwidth Usage

The Providers bandwidth usage graph illustrates inbound, outbound, and total bandwidth details on the same page as per a specific IRP instance. It allows cross-comparison between providers. The chart includes options to hide/show providers of interest during a selected period via the graph legend.

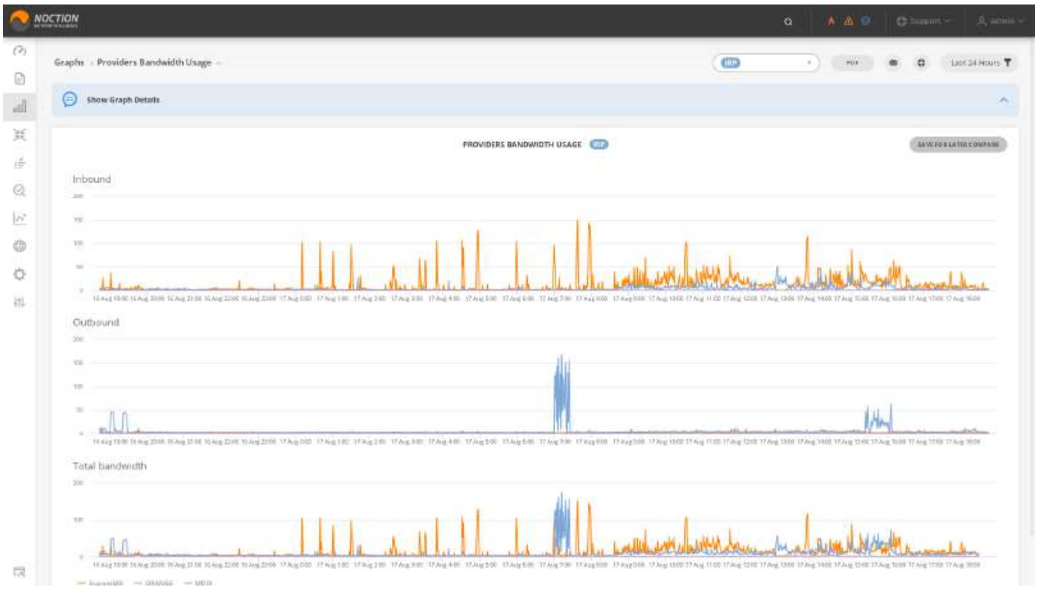

Figure 3.5.26: Providers Bandwidth Usage

3.5.25 Providers Efficiency

The horizontal bar charts are the graphical representation of the specific measurement results (packet loss, latency, successful and failed probes) obtained by IRP while performing numerous probes via all the providers.

Note:

- The provider with the best packet loss data is always placed on top of the others
- Filtering by date range and providers allows reviewing of past data or average performance over longer time intervals as well as showing/hiding specific providers.

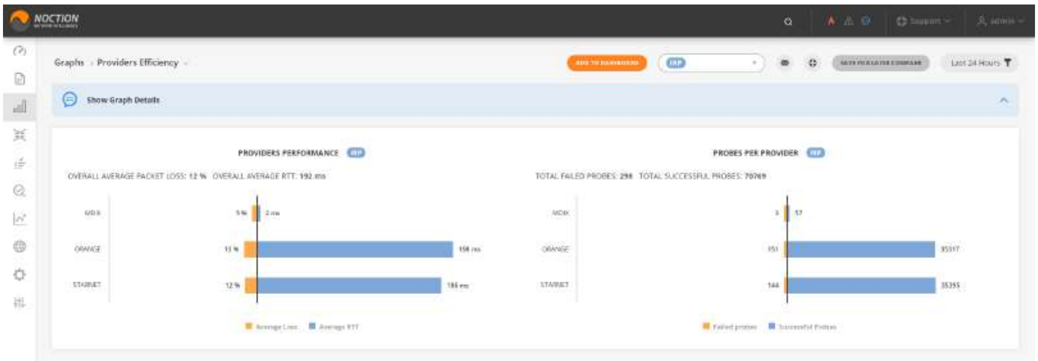

Figure 3.5.27: Providers Efficiency

While some providers seem better performing (for example, MDIX in the provided screen capture), the fact that they display a much smaller number of probes clearly indicates they are Internet Exchanges with only a few peers interchanging data with a given network.

A table view with exact details of the data is available in the report form as well.

See also: Providers Efficiency

3.5.26 Total and Improved Traffic

This graph allows users to see the positive impact of IRP instances on the network. It provides the amount of inbound/outbound improved traffic, as related to the total inbound/outbound traffic.

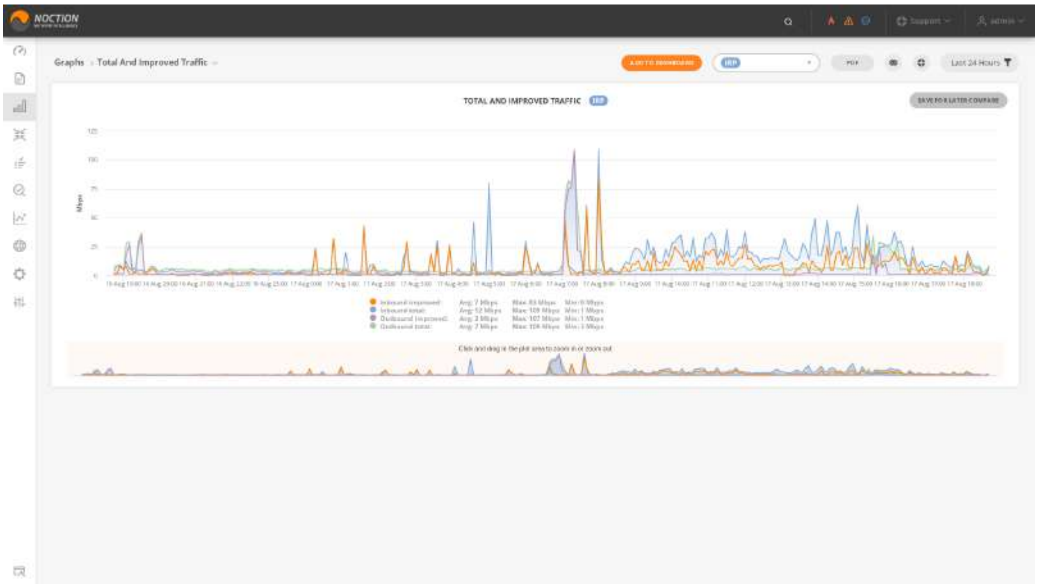

Figure 3.5.28: Total and Improved Traffic

3.5.27 Total Bandwidth Usage

The graph shows total inbound, outbound bandwidth as well as the In/Out 95th value as per specific IRP instance.

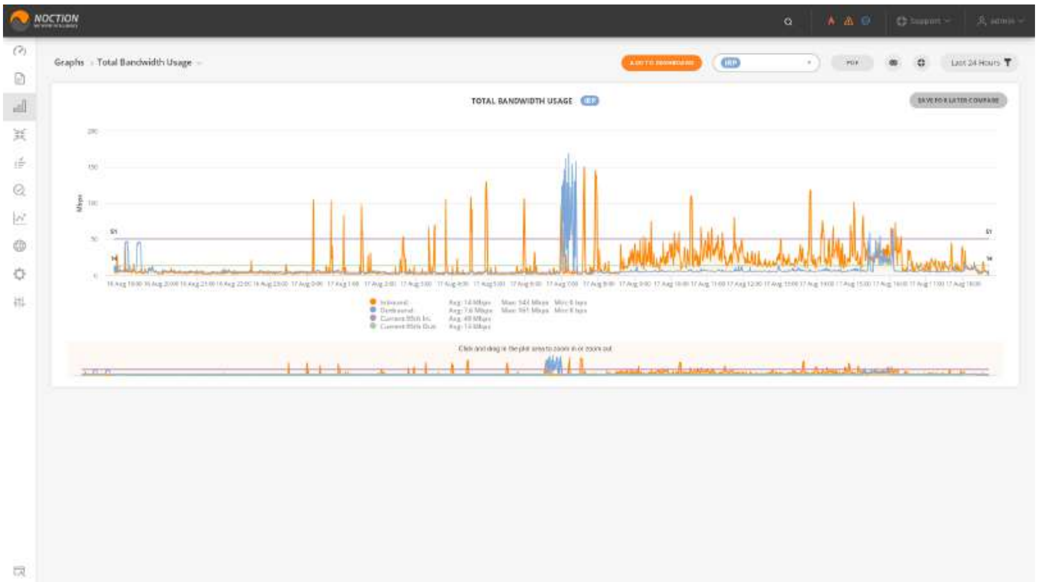

Figure 3.5.29: Total Bandwidth Usage

3.5.28 Unique Probed/Improved Prefixes

The "Unique Probed/Improved Prefixes" graph shows the number of improvements per time unit as reported to the total probed unique prefixes.

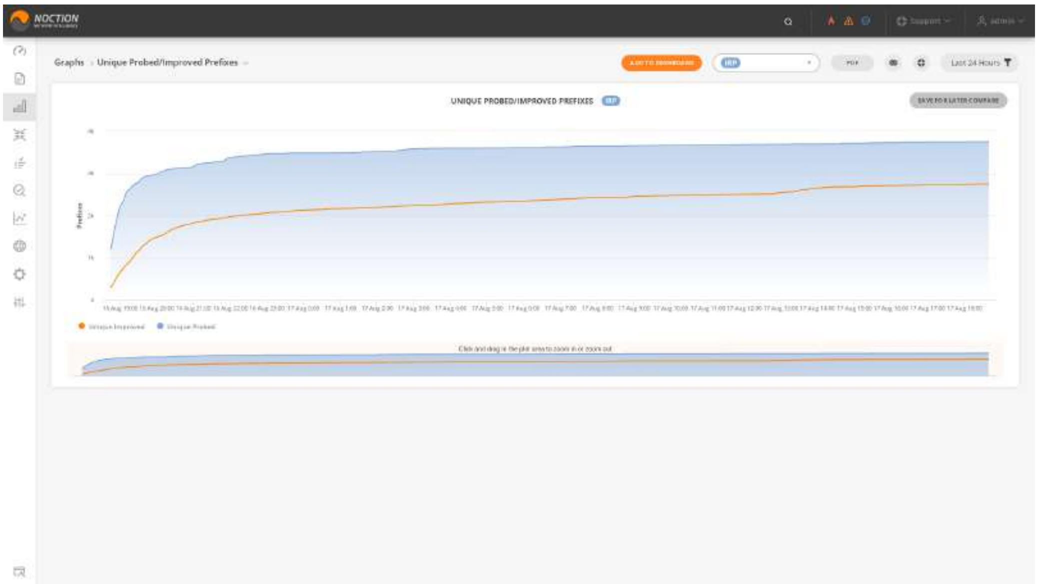

Figure 3.5.30: Unique Probed/Improved Prefixes

3.6 Inbound

The GMI's Inbound section is visually divided into two parts, each containing a set of links to reports, graphs, rules, and configuration options for both the Inbound Performance Optimization and the Inbound Commit Control features.

> The details on the Inbound Commit Control and Inbound Performance Optimization configuration are provided in the following sections: Inbound Commit Control and Inbound performance optimization configuration .

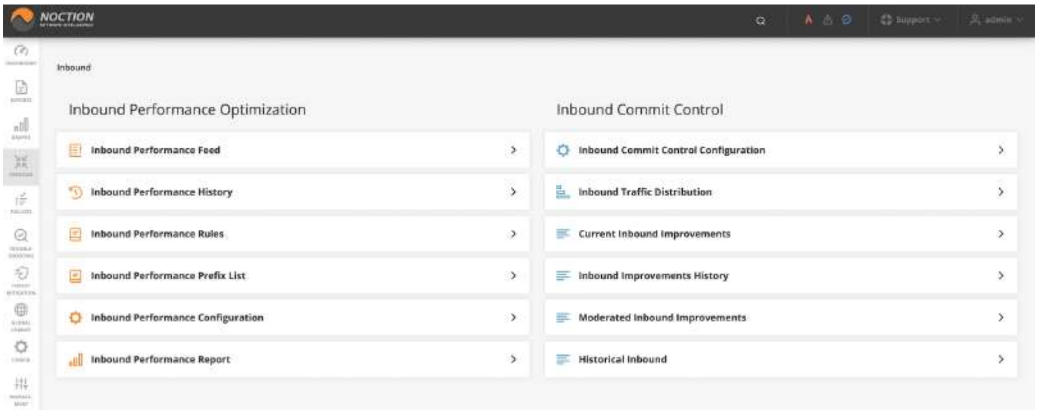

Figure 3.6.1: Inbound section

3.6.1 Inbound Commit Control

Starting with version 3.4 IRP introduced the Inbound bandwidth optimization feature, the configuration and reports for which are available in GMI.

3.6.1.1 Current Inbound Improvements

The “Current Inbound Improvements” report provides a list of the currently active improvements. The report shows the improved prefix and the providers from and to which the traffic was redirected. It also provides the reason, which the improvement was based upon.

If necessary, specific improvements can be manually deleted by clicking the check-box next to the needed prefix, afterward clicking the “Remove selected” button.

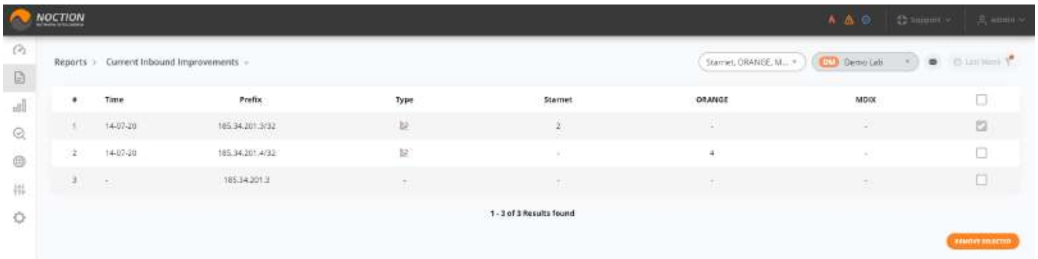

Figure 3.6.2: Current Inbound Improvements

3.6.1.2 Historical Inbound

The Historical Inbound traffic report provides an overview and analysis of the incoming traffic over a selected period of time per prefixes and the associated providers.

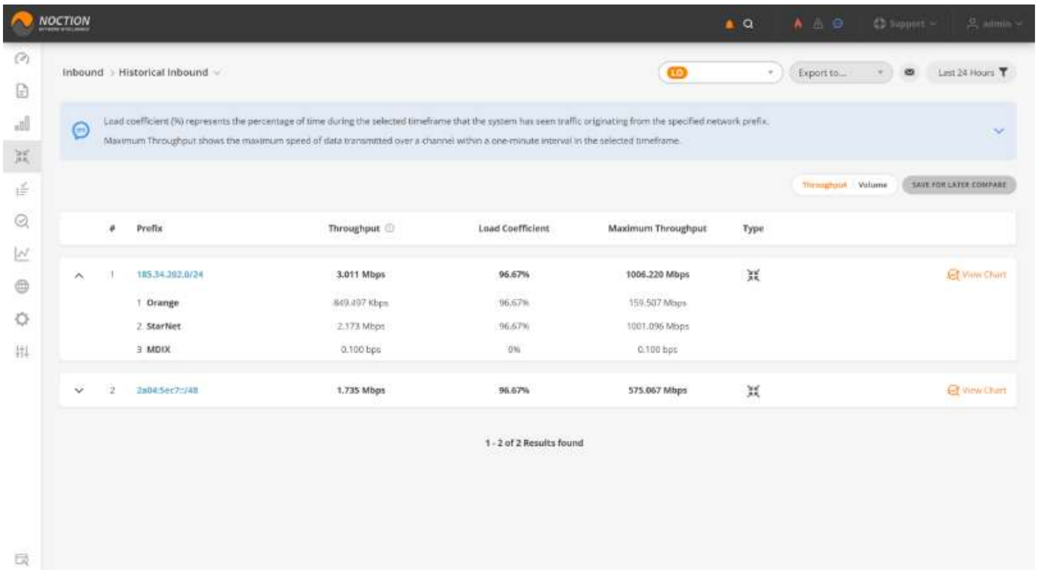

Figure 3.6.3: Historical Inbound

The report includes a range of relevant throughput and volume metrics related to inbound traffic, such as:

Prefix the network prefix associated with traffic

Throughput the average rate at which data is transmitted over a given period of time for a specific prefix

Load coefficient the percentage of time during the selected timeframe that the system has seen traffic per the specified network prefix

Maximum throughput shows the maximum speed of data transmitted over a channel within a one-minute interval in the selected timeframe

Volume this metric denotes the total amount of data transmitted or received for a specific prefix during the analyzed period

Count of updates the number of updates or changes made to the traffic data related to a specific prefix

Maximum volume similar to maximum throughput, this metric indicates the highest volume of data transmitted or received for a specific prefix during the analyzed period

Type categorizes traffic based on its direction. Inbound refers to data coming into a network from external sources, while Transit represents data passing through the network without being the final destination

View Chart displays a graphical representation of the throughput or volume of the inbound traffic for each specific prefix.

3.6.1.3 Inbound Improvements History

The "Inbound Improvements History" report provides a detailed list of past inbound improvement actions. The report shows the improved prefix and the new prepends count towards a provider.

Figure 3.6.4: Inbound Improvements History

3.6.1.4 Inbound Traffic Distribution

The graph highlights distribution of inbound traffic across inbound prefixes and providers. Inbound traffic distribution is available in the report form as well and includes a feature to cross-compare current values with a saved inbound traffic distribution from the recent past.

Figure 3.6.5: Inbound Traffic Distribution

3.6.1.5 Moderated Inbound Improvements

IRP includes a moderation feature that allows review and approval of Inbound improvements in GMI. If this feature is enabled, Inbound improvements are not automatically announced but instead, they are queued for review. If an inbound improvement has not been approved in time for a subsequent cycle of improvements an IRP instance will review the proposal itself and will adjust as needed thus making another recommendation.

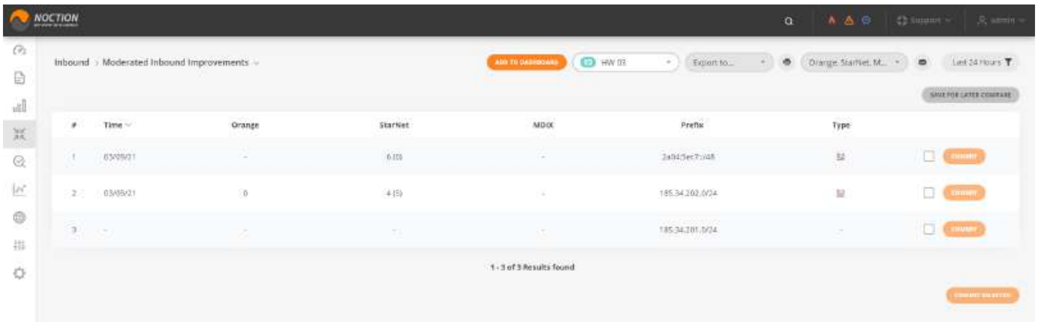

Figure 3.6.6: Moderated Inbound Improvements

Inbound Improvements moderation highlights the number of additional prepends proposed by IRP and allows submitting or rejecting individual improvements as well as a batch of selected improvements.

3.6.2 Inbound Performance Optimization

The Inbound Performance Optimization capability was added to the Intelligent Routing Platform starting with version 4.0. The feature is based on IRP's automated network performance metrics identification algorithm and the use of customer traffic engineering (TE) capabilities of the client's providers.

How Inbound performance works:Inbound performance optimization

3.6.2.1 Inbound Performance Feed

Inbound Performance Feed displays a list of inbound network performance improvements (per single IRP instance) that have been suggested, automatically applied, or manually enabled by a GMI user according to the pre-established set of Inbound Optimization rules. Each active/suggested inbound improvement entry contains the following details:

Status indicating whether the improvement is active/inactive

Improvement Type indicating if the improvement is Automated (suggested and activated by the system), Moderated (suggested by the system but not yet enabled by the user) or Manual (manually enabled by the user)

Description contains the name of the rule which triggered a particular action

TE Mark represents the Traffic Engineering community used within a particular rule

AS Path Match, BGP Community or Prefix List offers details on the matching criteria of the prefixes that are due to be improved as per the rule

Confirmation shows the time until the next improvement actuality verification

Initial Stats displays the packet loss and latency values for the provider in the rule before the improvement

Improvement Results displays the packet loss and latency values for the provider in the rule after the improvement activation

Enabled On displays the time and date the improvement was activated

Actions provides the options to enable/disable or completely delete an individual improvement.

> The average metrics shown might not reflect the computational algorithm logic and are approximate for the 'Improvement Results' and 'Initial Stats' fields.

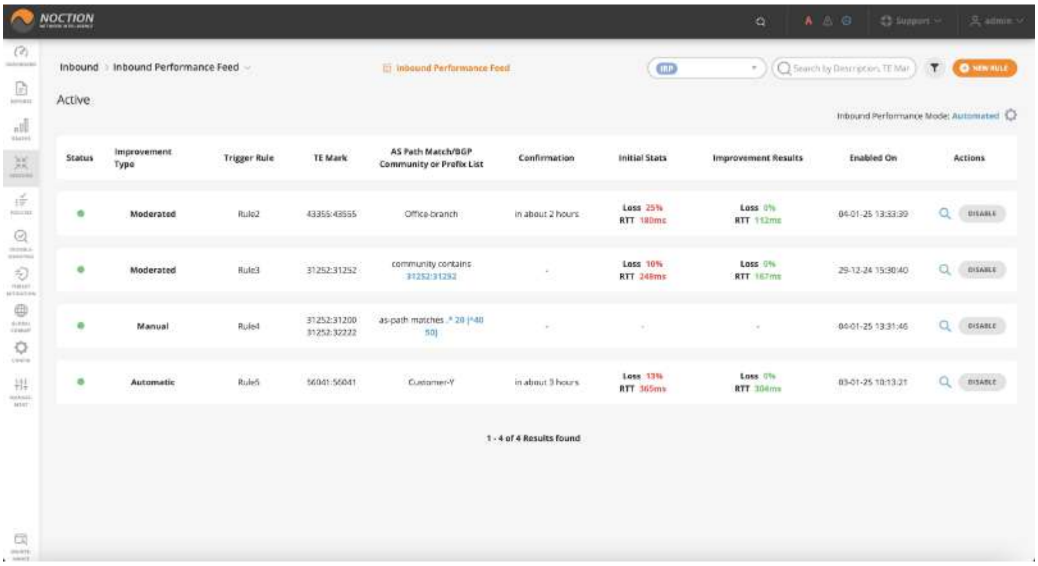

Figure 3.6.7: Inbound Performance Feed

3.6.2.2 Inbound Performance History

The Inbound Performance History tab displays a list of past (non-actual) improvements, with details of the time and date the improvement was deactivated.

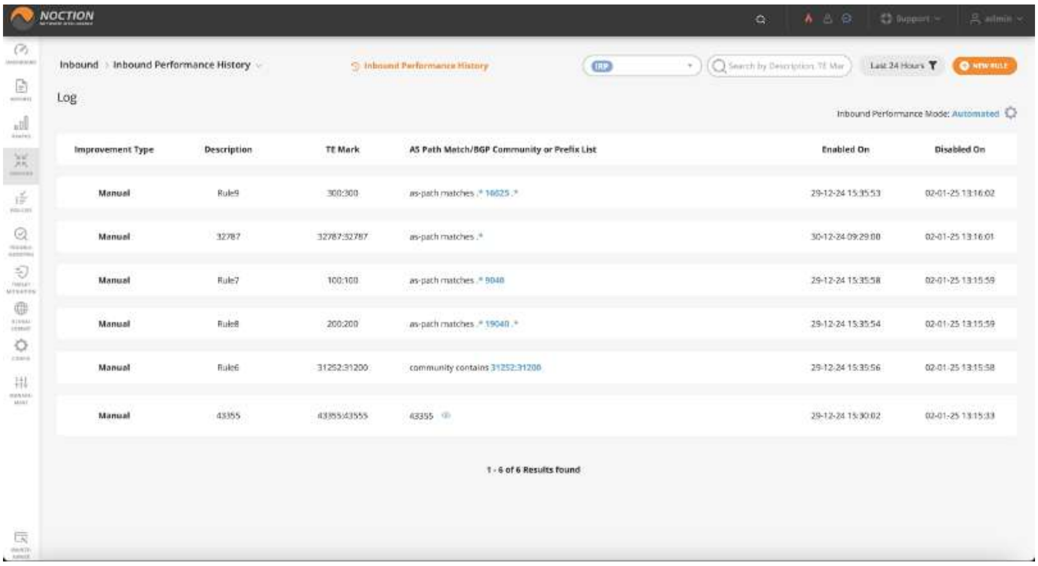

Figure 3.6.8: Inbound Performance History

3.6.2.3 Inbound Performance Prefix List

To enhance and refine inbound performance optimization, the ability to create and manage multiple prefix lists as matching criteria for inbound performance rules has been introduced. For more details on inbound performance rules, refer to section Inbound Performance Rules. The Inbound Performance Prefix List page allows users to create, edit, and delete prefix lists, as well as search for rules that use prefix lists as matching criteria.

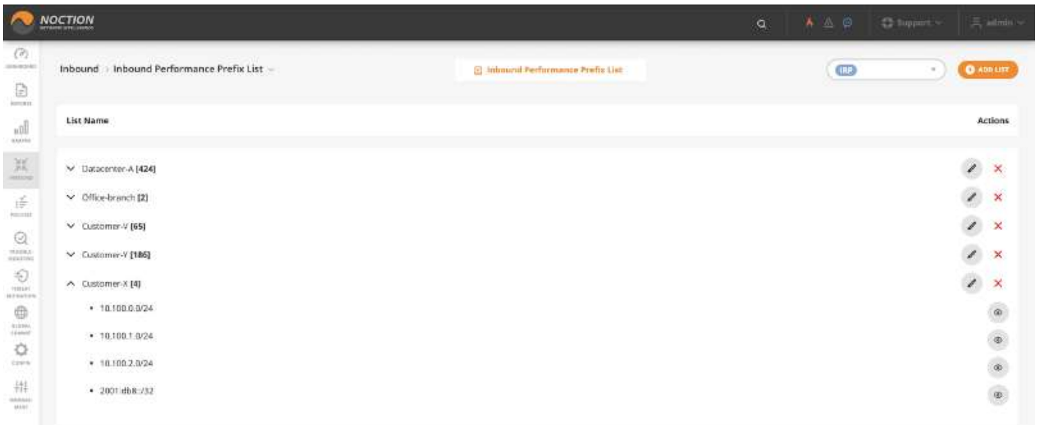

Figure 3.6.9: Inbound Performance Prefix List

To create a prefix list, click the ADD LIST button. Introduce details in the corresponding fields:

- Name
- List of prefixes

3.6.2.4 Inbound Performance Rules

Inbound Performance Rules tab displays a list of the inbound network performance rules (per single IRP instance) created by a GMI user according to the user's network specific needs and requirements. Each rule can be turned on/off, to enable or disable the subsequent validation of inbound traffic characteristics and the detection of possible inbound routing improvements. There are also options to instantly apply the improvement as per the rule manually, edit or delete it.

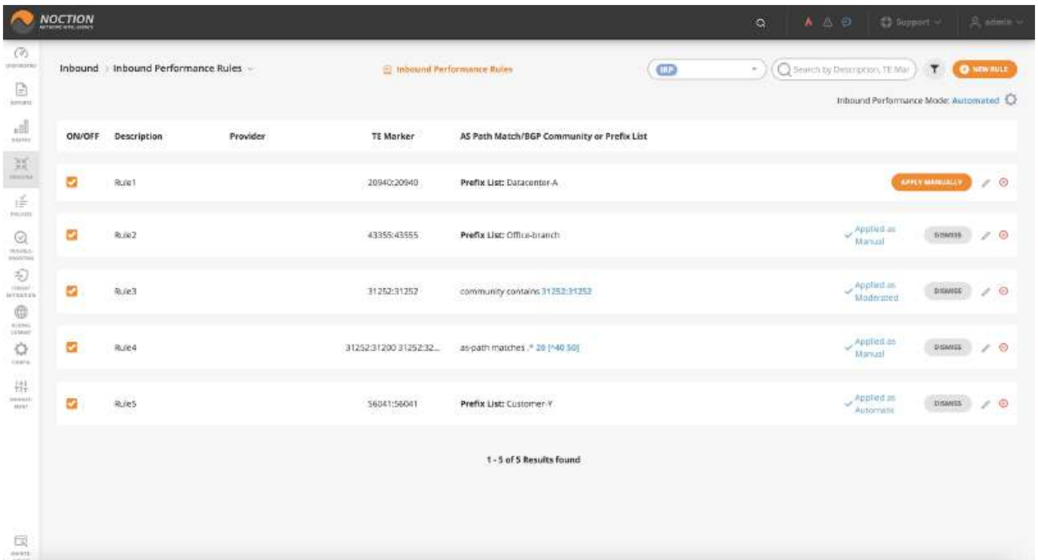

Figure 3.6.10: Inbound Performance Rules

To create a rule, click the NEW RULE button. Introduce details in the corresponding fields:

- **Description** - introduce the meaningful details for the rule you are about to create
- **Provider -** select the provider that the rule will apply to
- **Provider TE marker -** indicate the traffic engineering marker (a list of BGP communities) which has special meaning in context of traffic engineering capabilities of a specific provider

- **Improvement Activation Threshold (Packet Loss) -** indicate the packet loss value threshold (% delta worsening from the initial statistical model values)
- **Improvement Activation Threshold (Packet Latency) -** indicate the packet latency value threshold (% delta worsening from the initial statistical model values)
- **AS Path Match / BGP Community / Prefix List** (refer to Inbound Performance Prefix List) specify the selection criteria (**filter expression** or prefix list) of the prefixes that are due to be improved as per the rule

Filter expression

Filter expression is a boolean expression of sub-expressions:

- `as-path matches as-path-regex`
  for example: `as-path matches 1 2 10-20 [40 50-60]+ (. 200)*`
- `community contains bgp-community`
  for example: `community contains 55:66`

Boolean expression supports just 'and' and 'or' operators.

For example, filter expression "`as-path matches .* 20 [^40 50] and community contains 55:66`" would match an external prefix, the RIBIN record for which denotes:

- as-path "90 20 60"
- community set "2:3 55:66 40:50"

As-path regex

As-path regex implements subset of juniper as path regex, relaxing group operator '()' to support any expression within it. EBNF grammar:

```
regex = expr;
expr = seq | subexpr;
subexpr = repeat | term | group;
seq = (subexpr, seq) | subexpr;
repeat = ((group | term), "*")
| ((group | term), "+")
| ((group | term), "{", uint8, ",", uint8, "}";
any_of = "[", {range | atom}-, "]";
none_of = "[^", {range | atom}-, "]";
group = "(", expr, ")";
term = range | atom | wildcard | any_of | none_of;
wildcard = ".";
range = atom, "-", atom;
atom = uint32;
```

Examples

Regex matches input's start and end, as if "^regex$". If you need to skip numbers from start or end, use combination of 'repeat zero or more times' with 'wildcard', for example

- "9 5" matches only entire as-path "9 5", whereas
- ".* 9 5 .*" can match "1 2 10 9 5 30"
- Group and repeat on or more times operators "(. 5 6)+" can match "1 5 6 2 5 6 3 5 6"
- Repeat at least m and at most n times operator "2{3,4}" can match "2 2 2"
- Range "5-10" can match "6"
- Any of operator "5 [6 8]" can match "5 8"
- None of operator "5 [^6 8]" can match "5 9"

- The complex regex "5-10 [1 20-30]{1,2} (5 7{3,4})+ [^10]" can match "6 1 2 5 7 7 7 11", or "6 1 25 5 7 7 7 5 7 7 7 7 11", but not "6 1 25 5 7 7 7 10" because the path can't end with 10

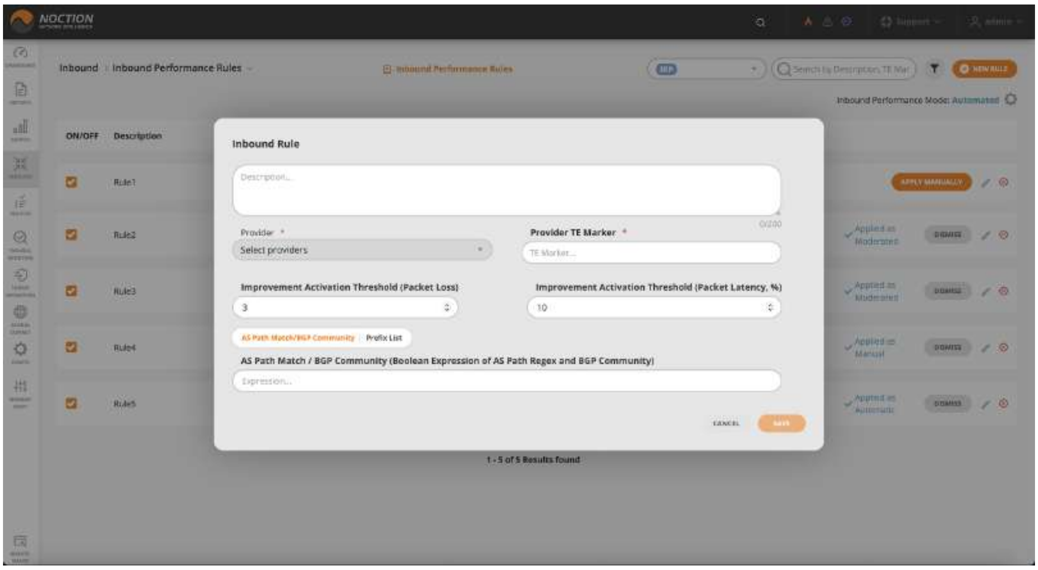

Figure 3.6.11: Creating an Inbound Performance Rule

Refer to: Inbound performance optimization configuration for the Inbound Performance configurations.

3.6.2.5 Inbound Performance Report

Inbound Performance Report contains two graphs showing the effect of the inbound improvements on bandwidth as well as packet loss and packet latency values. The presented data is displayed in accordance with a single inbound rule and IRP instance selected.

Figure 3.6.12: Inbound Performance Report Bandwidth Graph

Figure 3.6.13: Inbound Performance Report Loss/Latency Graph

3.7 Routing Policies

To see the full list of routing policies configured for a particular IRP instance, select the Policies option from the main menu and click on the Routing Policies option. The list displays all the enabled and disabled routing policies.

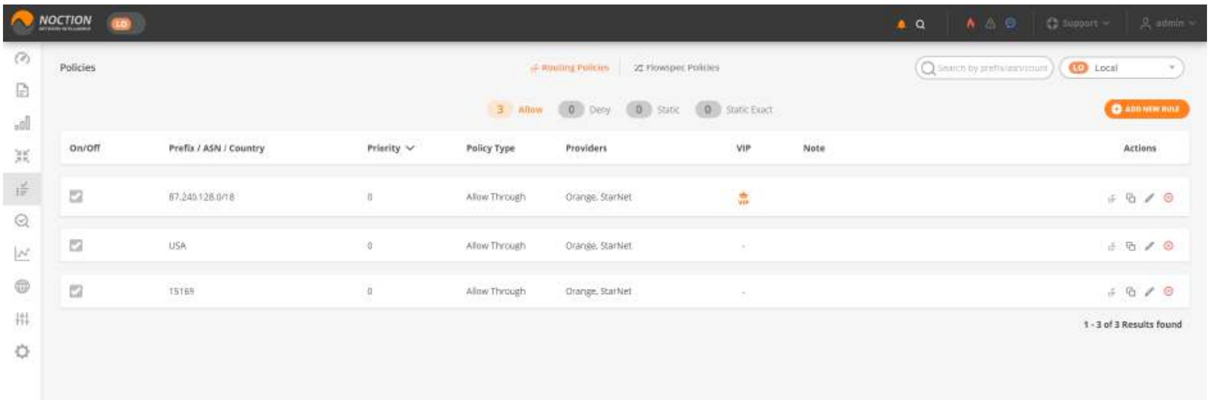

Figure 3.7.1: Routing Policies

As shown in the screenshot above, the list provides the following details:

- The ON/OFF state of each policy
- The country, prefix or ASN to which the policy applies
- Priority of a policy

> Priority specifies what policy to use in case of overlapping prefixes as might be the case of a large aggregate prefix policy extending over an ASN policy

- The type of the policy (allow/deny/static/static_exact)
- Providers which are affected by the policy
- The status of the VIP reprobing (enabled/disabled)
- A note to describe the policy
- The available actions for the policy. These include:
  - Show detailed routing policy information

  - Duplicate the routing policy
  - Edit the routing policy
  - Remove the routing policy

When opening up the detailed routing policy information, the following details become available:

- Time passed after the last probe was performed towards this prefix
- The time left before the next scheduled probe towards this prefix
- From and To details of an existing improvement relating to this policy.

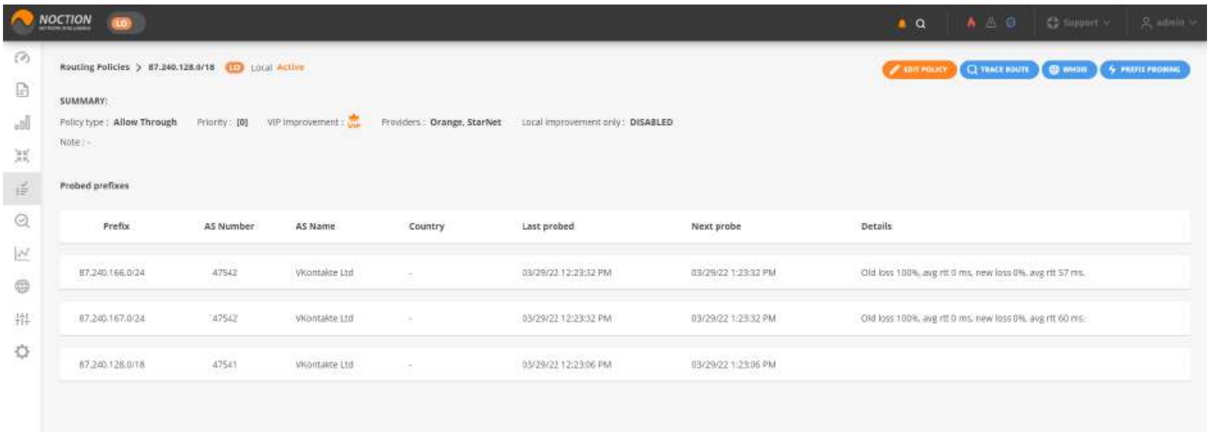

Figure 3.7.2: Detailed routing policy view

A new routing policy can be added by pressing the "ADD NEW RULE" button. The following window will open up:

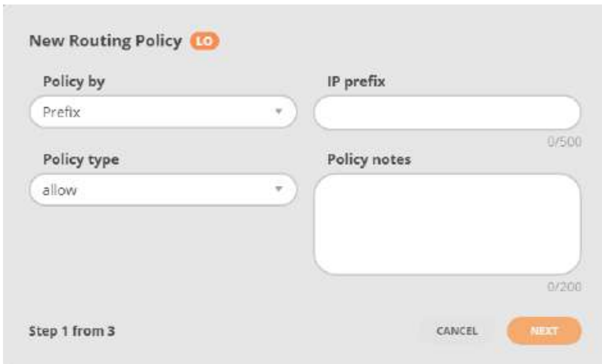

Figure 3.7.3: Routing policy window

Depending on the “Policy by” field selection, some of the following parameters should be configured to add a routing policy:

- Prefix: The prefix to which the policy is applied
- ASN: The ASN to which the policy is applied
- Country: The country to which the policy is applied. Such a policy applies to all prefixes known to be located in the country.
- Note: A note to describe the policy
- Policy Type: The type of the policy (allow/deny/static/static_exact)

- A community to mark improvements
- A flag to set if policy is VIP
- The policy status (enabled/disabled)
- A flag to indicate if an ASN policy should propagate to downstream ASN.
- A flag indicating if a global improvement is allowed.
- Priority slider or text-box will set a policy's priority.
- The Providers for which the policy should be respected
- Cascade flag for ASN policies that indicate if the policy should be enforced only for this AS or also for downstream AS. If cascading is enabled it ensures that the policy is enforced for all traffic that will eventually be routed through this AS.

Refer policy configuration parameters Routing Policy for details.
Click any of the "Allow", "Deny", "Static" or "Static Exact" options to see a list of policies of a particular type.

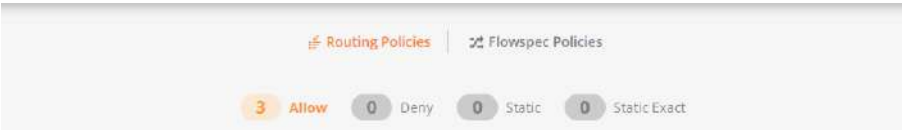

Figure 3.7.4: Routing Policy Type

3.8 Flowspec Policies

To review Flowspec policies, select the Routing Policies option from the main menu and proceed to the Flowspec Policies tab.

> ⚠ Flowspec must be enabled globally for each IRP instance under Configuration > Global as well as each router under BGP > Router Name > Advanced tab.

The list displays all the enabled and disabled Flowspec policies configured for a particular IRP instance. Click any of the "Redirect", "Throttle", "Drop" or "Redirect IP" options to see a list of policies of a particular type. Flowspec policies are also grouped as follows: Prefix Policies, ASN Policies, Country Policies, and Other Policies (Protocol/DSCP).

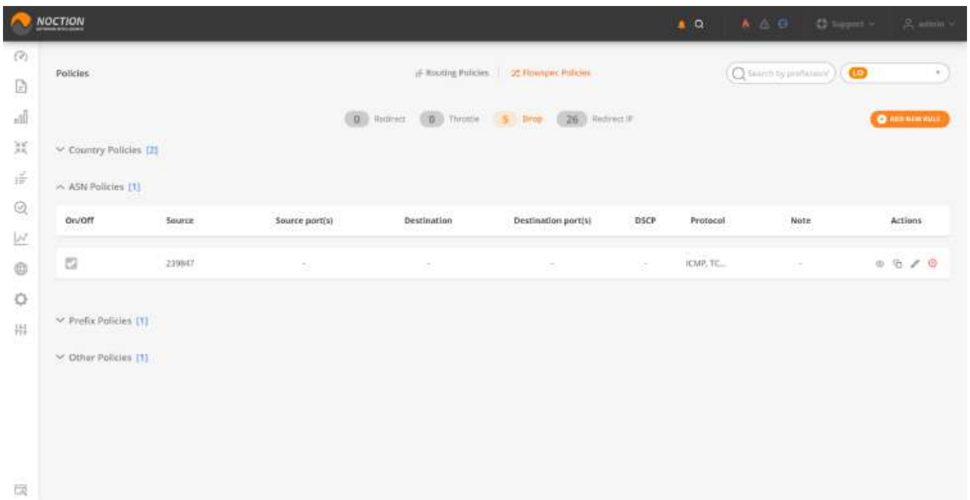

Figure 3.8.1: Flowspec policies

Depending on the selected policy type tab, the list highlights:

- The ON/OFF state of each policy
- A source ASN/prefix/country and port(s) for matching packets
- A destination prefix and port(s) for matching packets
- DSCP traffic classification value
- Protocols of matching packets, e.g., TCP, UDP, or ICMP
- Redirect IP or Provider for the redirect to VRF policies
- A note to describe the policy
- The available actions for the policy:
  - Display detailed information about the FlowSpec policy
  - Duplicate the Flowspec policy
  - Edit the Flowspec policy
  - Remove the Flowspec policy

A Flowspec policy is added by clicking on the designated "ADD NEW RULE" button.

Figure 3.8.2: Flowspec policy popup

The first step prompts you to choose the IRP instance as well as the type of Flowspec policy you are about to create. There are 4 options:

- Policy by Prefix
- Policy by ASN
- Policy by Country
- Protocol/DSCP policy

Depending on the selection, some of the following parameters should be configured:

- Source Prefix/ASN/Country/Port(s): The source ASN/prefix and port(s) of the IP packets that match. A prefix in CIDR notation or a single IP address should be provided. Multiple valid TCP/UDP ports can be provided, as well as port ranges.

ASN/Country Flowspec policies can be set up only on the source of IP packets

Ensure that routers are capable of processing a large number of Flowspec rules before setting policies for an AS/Country consisting of a very large number of network segments (prefixes)

- Destination Prefix/Port: The destination prefix/port attribute of the IP packets that match. Same rules as for Source Prefix/Port(s) apply
- Protocols: packet protocols that match the policy. Can be filtered down to one or a combination of the following protocols: TCP, UDP, ICMP
- DSCP traffic classification value
- Policy Type: The type of the policy (Throttle, Drop, Redirect, or Redirect IP)
- Provider specifies one of the provider identifiers where traffic will be redirected. The provider is set only for Redirect policies

In order for Redirect policies to be implemented on routers, an extended community value must be assigned to them, and VRF must be configured on the router itself. Consult Noction support for details

- Rate limits the allowed bandwidth usage for matching traffic. The value is set only for Throttling policies. The rate specifies a number in the range of 1-4200 Mbps
- Exempted Prefixes/ASNs are the lists excluded from country policies. The fields are relevant to the country policies only.

When creating an ASN Flowspec rule and selecting the Throttle policy type, the value introduced under "Rate limit" applies to individual prefixes within the AS, and not the Autonomous System as a whole

If the provided Flowspec policy attributes are incomplete or invalid on attempting to submit it, a warning will be raised

For more details, refer to Flowspec configuration parameters, for example: global.flowspec, core.flowspec.max, core.flowspec.max_ipv6, bgpd.peer.X.flowspec, peer.X.flowspec.ipv4.redirect_community, peer.X.flowspec.ipv6.redirect_community.

3.8.1 Policies by Country

Policies by Country, as part of the Flowspec Policies functionality, provides IRP users with the geo-blocking capability at the BGP level. Such policies allow network operators to restrict internet traffic by manipulating routing decisions based on geographic regions, particularly countries. Network administrators can define packet filtering rules based on additional parameters, such as protocols, port numbers, and destination prefixes, enabling a more granular approach. Moreover, users can access lists of affected prefixes and ASNs associated with each specific country for every policy, facilitating more informed routing decisions.

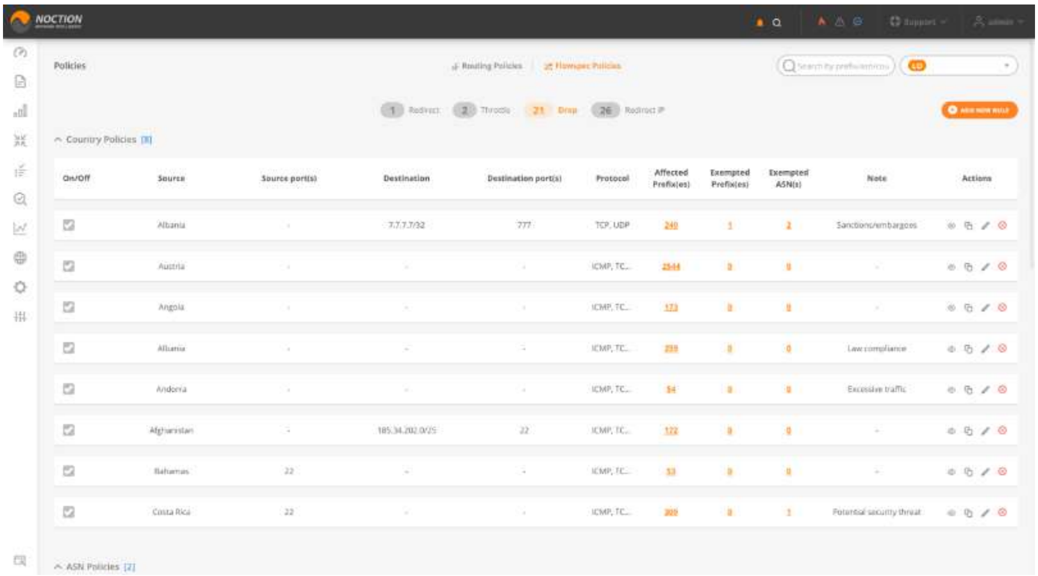

Figure 3.8.3: Policies by Country

Once enabled, each policy contains statistics on the number of affected prefixes that can be viewed for specific details on the actual list of prefixes as well as ASNs that the prefixes belong to. The history of changes is maintained for each policy.

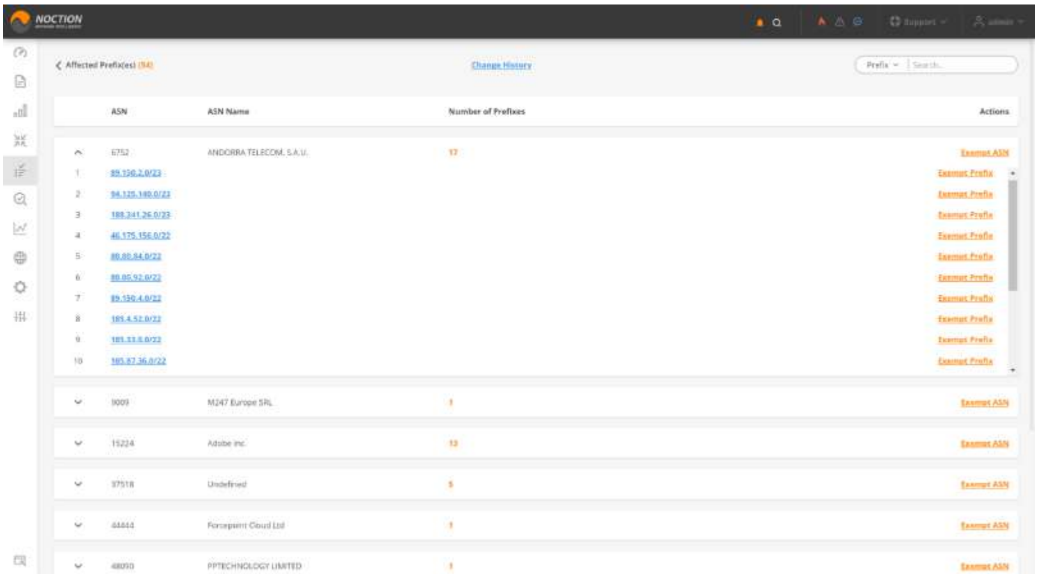

Figure 3.8.4: Policies by Country - Affected Prefixes view

To avoid routers overflowing with multiple policies, there are some limitations to the number of Country/ASN policies one can create. A maximum of three policies for a particular Country/ASN can be created, with each of them being different from the other two by having an IPv4 destination prefix, an IPv6 destination prefix, or no destination prefix at all

3.9 Troubleshooting

The GMI Frontend provides several troubleshooting tools that can offer you quick information on the specific remote networks.

3.9.1 Looking glass

A ping, traceroute or MTR can be run from the GMI Frontend via different providers available in particular IRP instances. The results are displayed on the same page in specific query results sections that can be expanded.

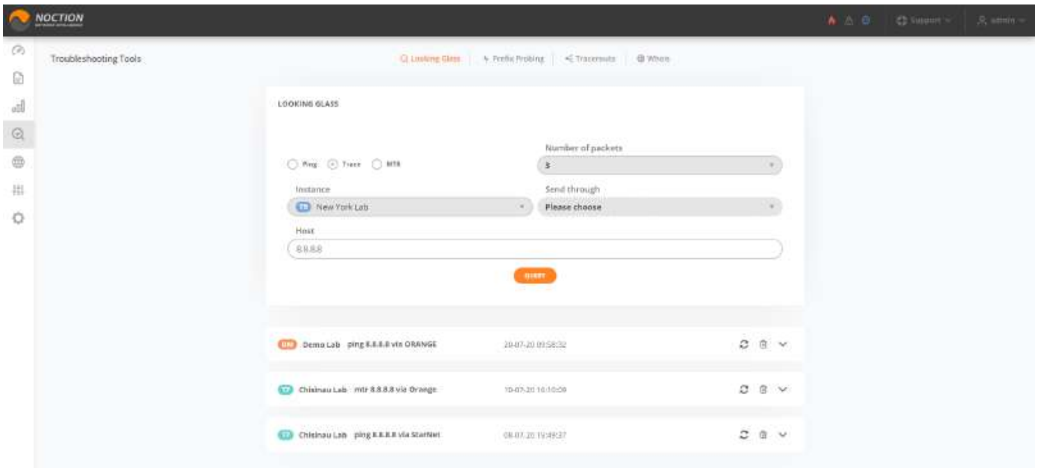

Figure 3.9.1: Looking glass

3.9.2 Prefix probing

GMI features a manual prefix probing function. One can manually submit a specific IP address for probing and optimization by particular IRP instances. Valid hostnames and IP addresses can be entered. The probing results will be displayed on the same page. Please note that the selected IRP instances probes the exact entered IP address.

In case the IP address doesn't respond, an IRP instance runs the indirect probing process and optimizes the whole prefix based on the indirect probing results.

Prefix probing allows to automatically or manually choose the best path for a particular IP address or prefix.

In order to check what are the possible paths for a prefix, enter the IP address or prefix into the field. Checkmark the "Fast Probing" option to send only a small batch of 5 control packets (instead of a 100 packets full probe) to establish performance characteristics through different providers. If you prefer the prefix to be improved automatically, tick the check box "Improve automatically" and press the "Submit for probing" button. Otherwise, just press the "Submit for probing" button and wait until the system returns the probing results.

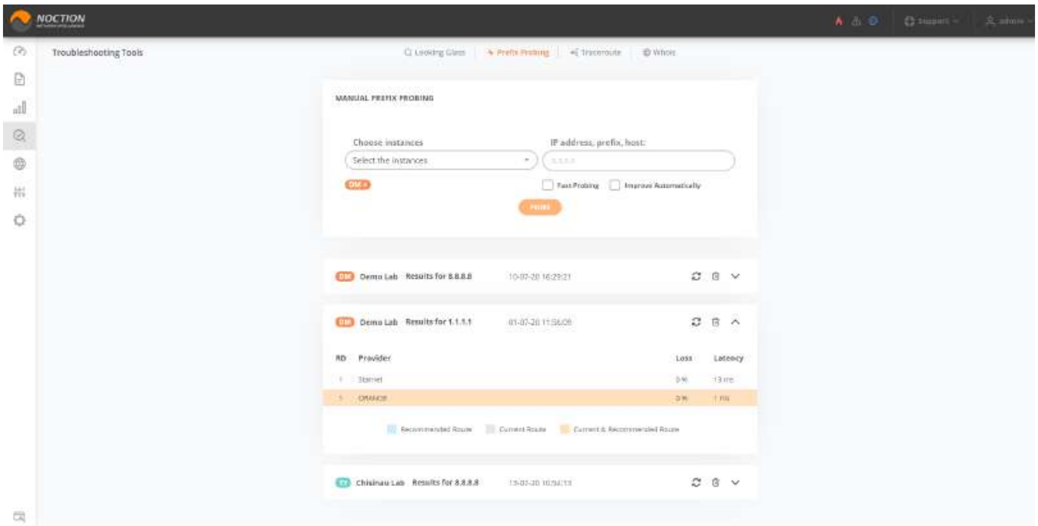

Figure 3.9.2: Prefix Probing

The system returns the probing results by displaying the loss and latency values for each of the providers. After the probing results are received, choose the path you consider the best one. Note, that the current and the best path are highlighted.

3.9.3 Traceroute

A traceroute to a specific IP address or hostname can be run from any IRP instance. Results will be displayed in the graphical and detailed table formats, as in the examples below. The graph can be presented in full view and exported as an image.

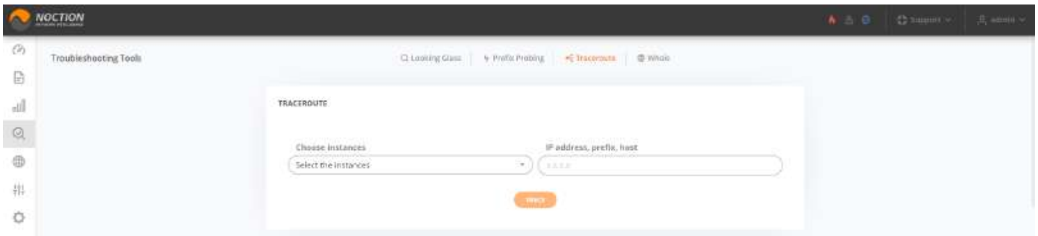

Figure 3.9.3: Traceroute

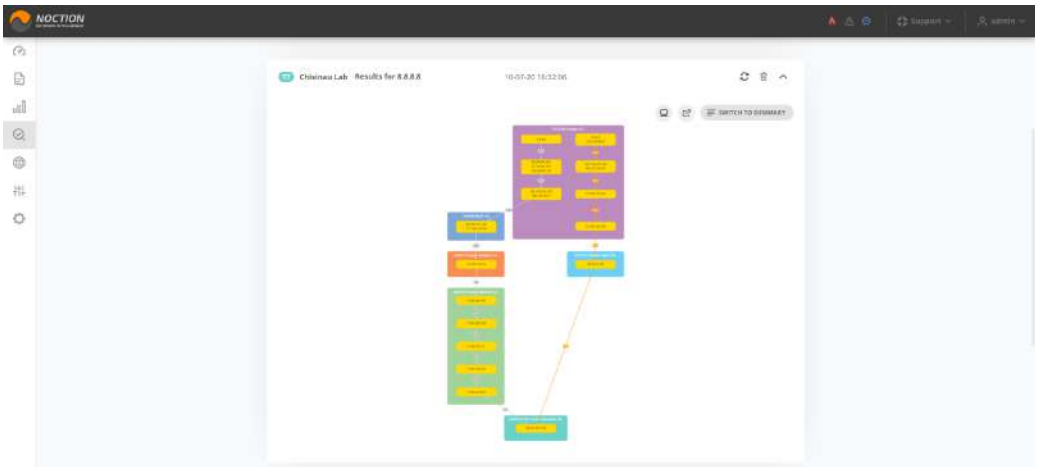

Figure 3.9.4: Traceroute results

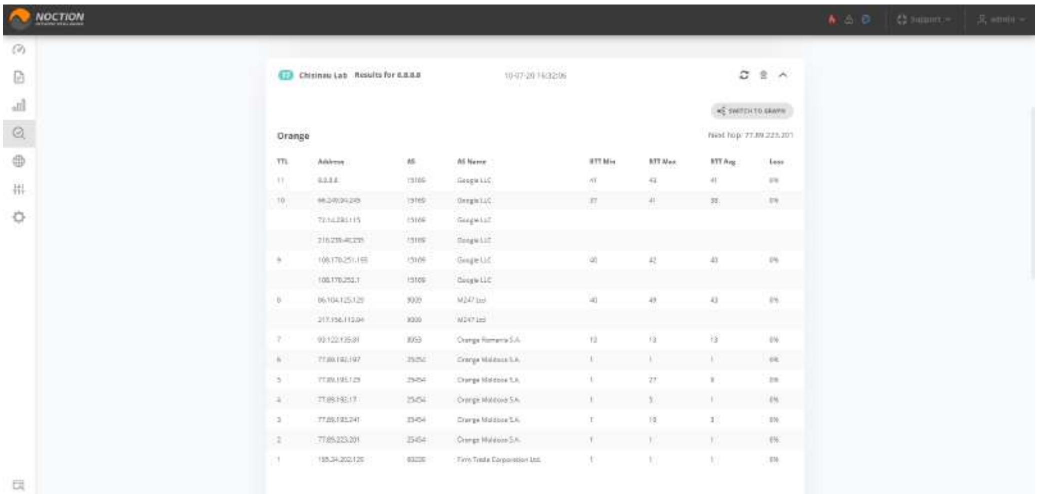

Figure 3.9.5: Traceroute exploring details results

3.9.4 Whois

The “Whois” tool can query IP addresses, hostnames, network prefixes, or AS Numbers. The following valid values can be entered in the search box:

- Hostname: fully qualified domain name, e.g: “domain.com”
- IP: valid IP address, e.g: “10.20.30.40”
- IP Block: any valid prefix, in CIDR format, e.g: “10.20.30.40/24”
- ASN: valid ASN, using the following format: “AS1234”

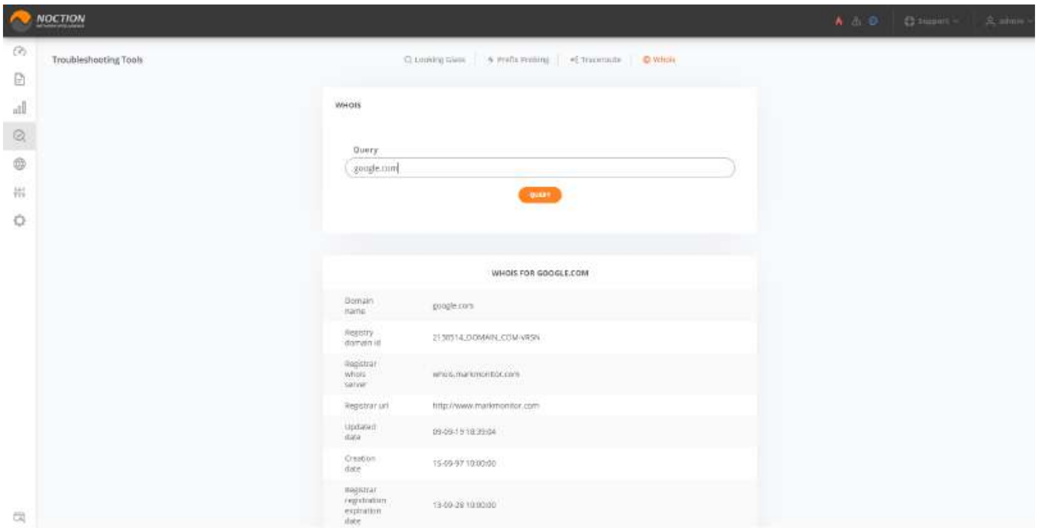

Figure 3.9.6: Whois

3.10 Threat Mitigation

The Threat Mitigation feature can be specifically used to better understand and automatically mitigate the effects of the Distributed Denial of Service (DDoS) attacks. The feature uses standard telemetry and control (NetFlow/sFlow and BGP) capabilities of routers to automatically block disruptive volumetric denial of service attacks. The solution leverages standard features of modern routing hardware to scale easily to large high-traffic networks. It employs a set of threshold-based user-defined rules for potential attack detection as well as the use of FlowSpec and Blackholing mechanisms to mitigate the detected attack.

3.10.1 Monitor

The Monitoring tab offers a 30-minute router(s) traffic graph view, as per specific IRP instance, providing the total Bytes and Packets statistics along with the filter down capability on packet types, including Syn, Fin, UDP, ICMP.

The graph gets updated automatically every minute. To pause the auto-update, click the data point of interest on the graph. This will populate the Historical Data table on the lower left side as well. Click the RESET button to enable the graph auto-update.

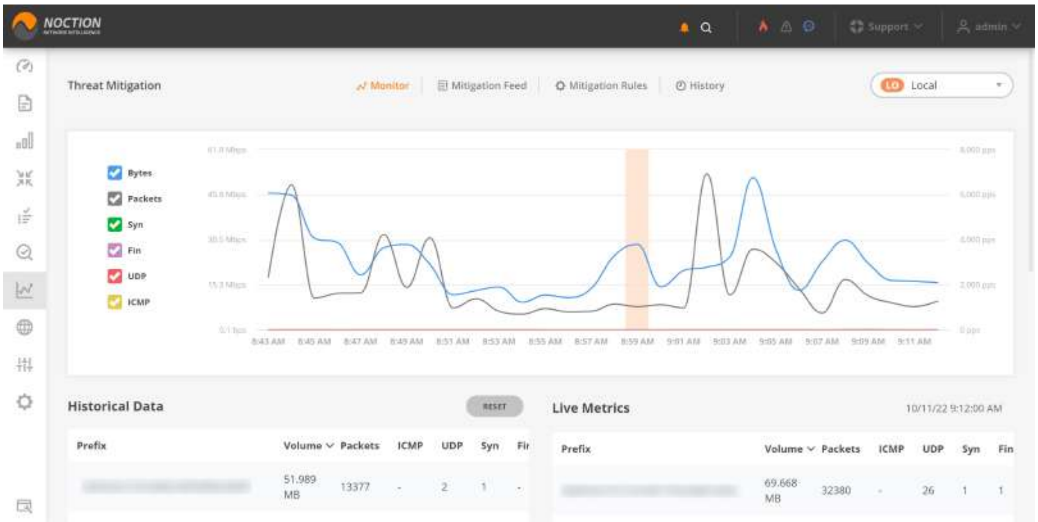

Figure 3.10.1: Threat Monitor

3.10.2 Mitigation Rules

All the configured rules are listed under the Mitigation Rules tab.

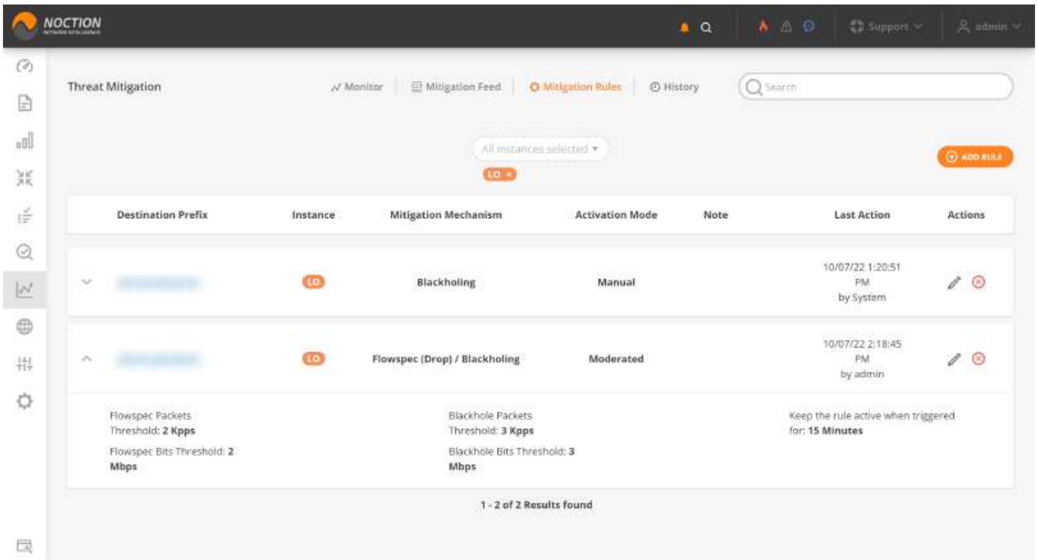

Figure 3.10.2: Mitigation Rules

To add the new rule, click the NEW RULE button. Fill out the required fields and hit CREATE. Based on the chosen Activation mode, a particular threat mitigation action will either be presented as a suggestion under the **Mitigation Feed** tab or started automatically.

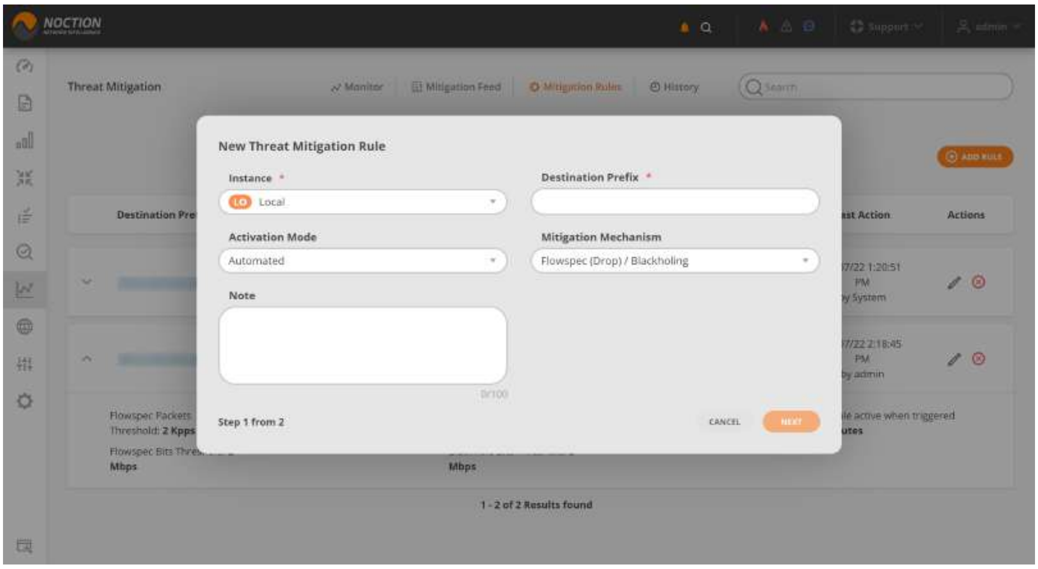

Figure 3.10.3: Add new rule

To be able to set up a specific Flowspec and/or Blackholing mitigation mechanism, details should be first provided at the Router and Provider level under the Configuration editor section.

In case the individual rule threshold values are not provided, the default values provided in the configuration section get used. Default values for Threat Mitigation rules are set in these configuration parameters: Threat mitigation

Deletion of an existing mitigation rule does not affect existing mitigation feed entries. They should be removed separately.

3.10.2.1 Mitigation Mechanisms

BGP Blackholing BGP Blackholing mechanism in Threat Mitigation allows network operators to selectively block or drop traffic to a specific prefix.

Once a rule with the corresponding mechanism gets triggered, the Blackholing community (peer.X.blackholing.community) is attached to an announced prefix by the Blackhole Routers (peer.X.blackholing.bgp_peer). The purpose here consists of signaling to a provider network that a neighbor router should discard all traffic destined for the received prefix with the attached BGP blackhole community.

It's worth noting that BGP Blackholing should be used with caution, as it will drop all the traffic to the blackholed prefix. This can cause serious consequences, such as dropping legitimate traffic. That's why it's advised to use it carefully and with a clear plan to avoid any collateral damage.

FlowSpec Drop The FlowSpec Drop mechanism allows network administrators to drop packets towards a destination prefix.

Once the FlowSpec Drop Threat Mitigation rule gets triggered, it is propagated to all the configured routers in IRP that support FlowSpec. When a router receives a FlowSpec drop rule, it is installed in its forwarding plane and used to match incoming packets. If a packet matches the rule, it is dropped and not forwarded to its destination.

BGP redirect The BGP redirect mechanism allows network administrators to manually or automatically tag a particular route with a BGP community value and subsequently signal client or provider routers to apply specific actions or policies to such routes.

To redirect traffic to a specific IP address or prefix using the BGP Redirect feature, a network administrator would first define a redirect community(s), then set up the custom threat mitigation rule(s), which, when triggered, would associate the community(s) with a route by adding it as an attribute to the route. Next, depending on the case, the network administrator would configure the router to apply a routing policy that matches the community value and redirects traffic to the desired IP address or readvertises it to the upstream providers.

It's important to note that, the use BGP redirect mechanism requires a good understanding of BGP protocol, network topology and routing policies.

FlowSpec Redirect IRP's Threat Mitigation feature offers the FlowSpec Redirect to IP mechanism, allowing network operators to automatically redirect specific traffic to an IP address for further processing or inspection.

During a DDoS attack, an attacker floods a targeted network or service with a large amount of traffic in an attempt to overload it and make it unavailable to legitimate users. To protect against this type of attack, network administrators can configure the default/custom rules and employ the FlowSpec redirect to IP mechanism to divert traffic that violates predefined thresholds and matches a particular destination prefix to a DDoS mitigation/scrubbing service.

The DDoS mitigation/scrubbing service can then analyze traffic and block or rate-limit the malicious traffic while allowing legitimate traffic to pass through. By redirecting traffic to a DDoS mitigation service using FlowSpec, network administrators can effectively protect their network and services from DDoS attacks without having to block all incoming traffic.

FlowSpec mitigation method can be chosen to filter out smaller attacks while the Remote Triggered Blackhole should be sent to providers to block large volume attacks. Both Flowspec and Blackholing mechanisms can be enabled for a single rule, with the thresholds for Flowspec being indicated lower than the Blackholing values. In this case, when the FlowSpec Mitigation is not capable of handling the attack the BGP Mitigation comes into play.

3.10.3 Mitigation Feed

The Mitigation Feed represents a list of currently active rules or suggested actions once a particular rule gets triggered. Users can accept the suggestions or simply delete any entry from the list.

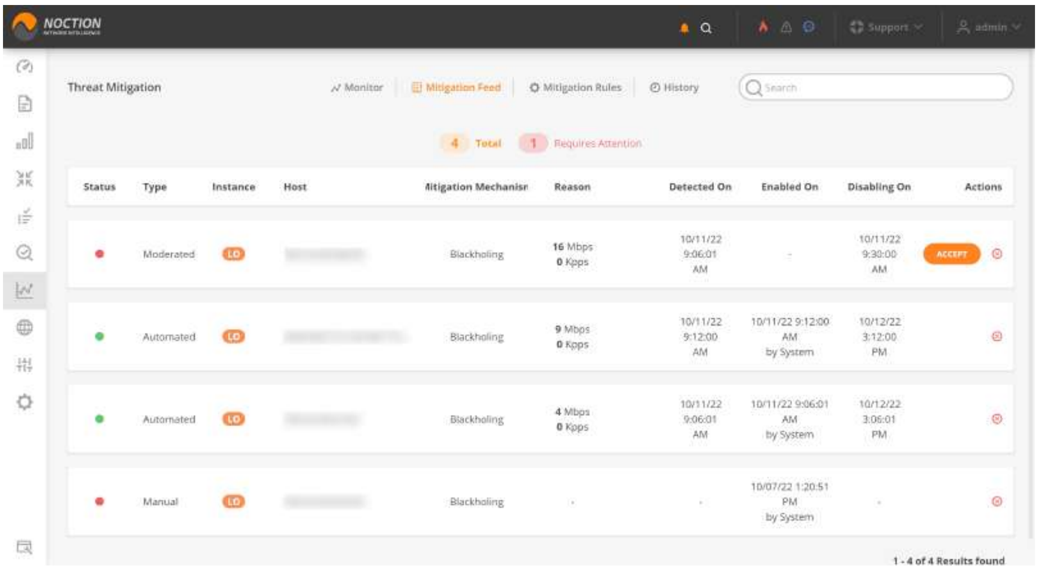

Figure 3.10.4: Mitigation feed

3.10.4 History

The "History" tab highlights the time and the past threat mitigation action details, facilitating the tracing of various issues on the network and in any particular rule configuration per each IRP instance.

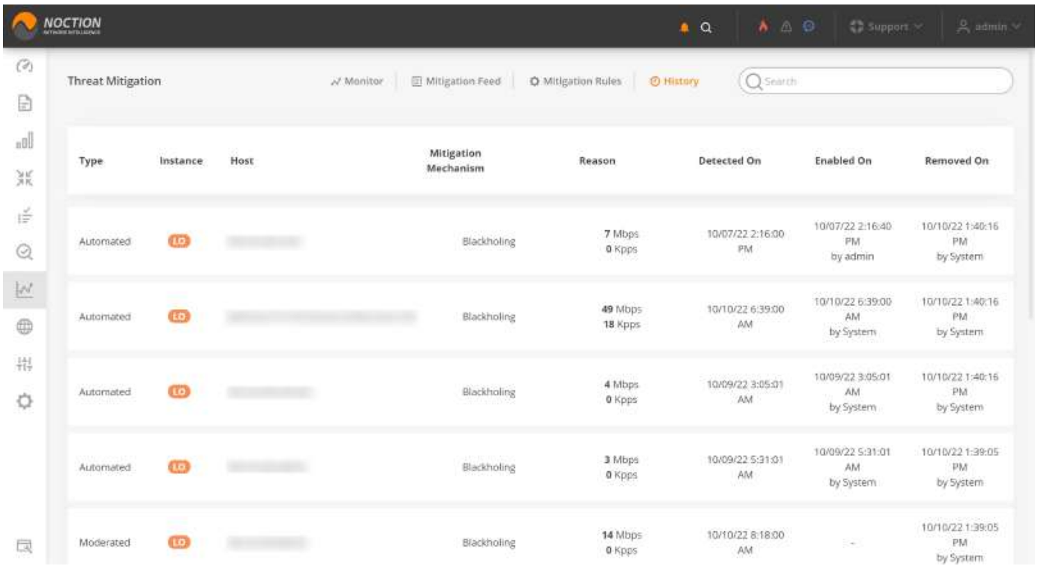

Figure 3.10.5: History

3.11 Global Commit

Customers can deploy network configurations with many actual links going to a single ISP from different points of presence. The additional links can serve various purposes such as to provision sufficient capacity in case of very large capacity requirements that cannot be fulfilled over a single link, to interconnect different points of presence on either customer (in a multiple routing domain configuration) or provider sides, or for redundancy purposes. Individually all these links are configured in IRP instances as separate providers. When the customer has an agreement with the ISP that imposes an overall limitation on bandwidth usage, these providers (from various IRP instances) can be grouped together in GMI so that the whole group can be optimized.

Global Commit options can be configured and results visualised by navigating to the corresponding tabs within the Global Commit section.

3.11.1 Global Commit Configuration

Go to the Configuration tab, drag and drop the providers of choice from the right side instances/providers options available to create Global groups. Providers will get distributed and marked by the IRP instance they belong to automatically within the Global group.

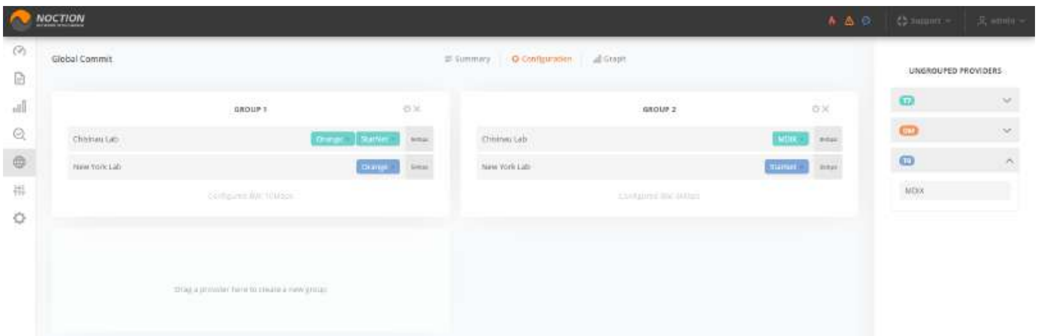

Figure 3.11.1: Global Commit groups configuration

Next, provide the "Global Group local member 95th percentile" (95th for the providers from a specific IRP instance within a Global group) as well as "BW reserved for local instance operation" (the amount of summed up bandwidth reserved to providers on specific IRP instance) values.

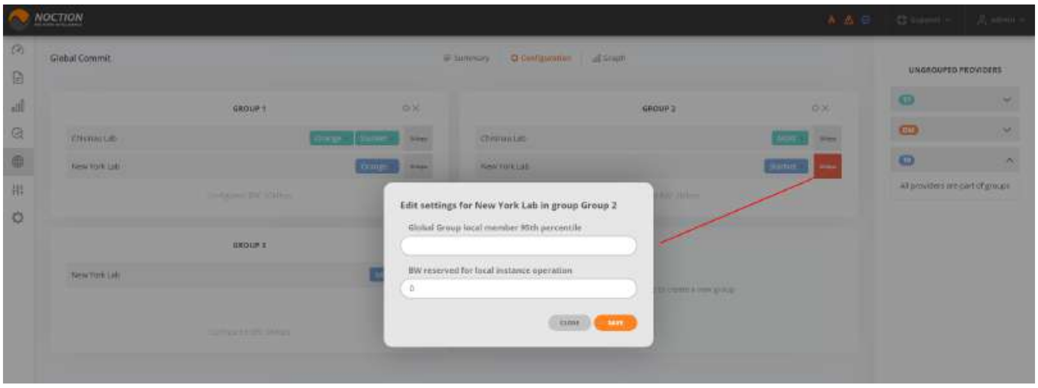

Figure 3.11.2: IRP instance individual settings within the Global Group

Click on the Global Group gear icon to change the group name, add the global group's members (hosts), Global Group high load bandwidth limit parameter similar in function to (4.8.25) and Global Group low load bandwidth limit parameter similar to (4.8.26)

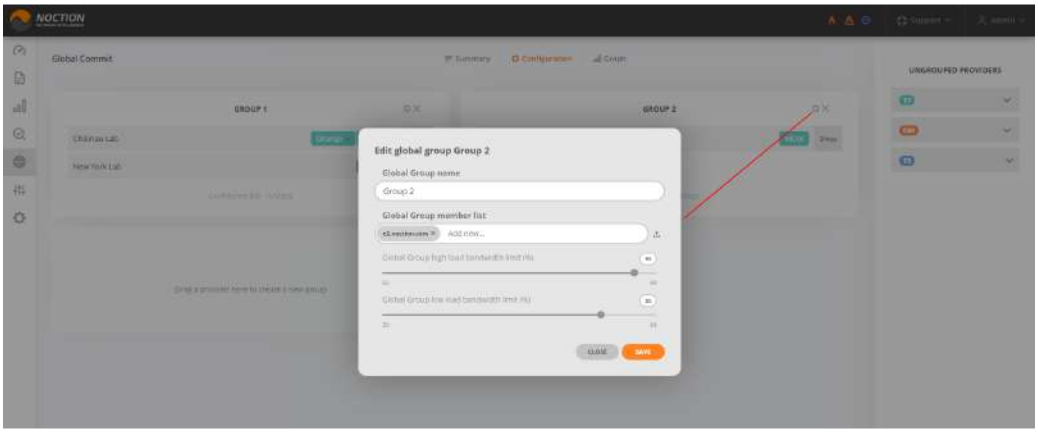

Figure 3.11.3: Global Commit Group Settings

3.11.2 Global Commit Summary

This tab offers current bandwidth statistics for Global Groups as well as instances and providers within groups. The configured 95th and Reserve Bandwidth values as well as a refresh and graph view link is offered for each group.

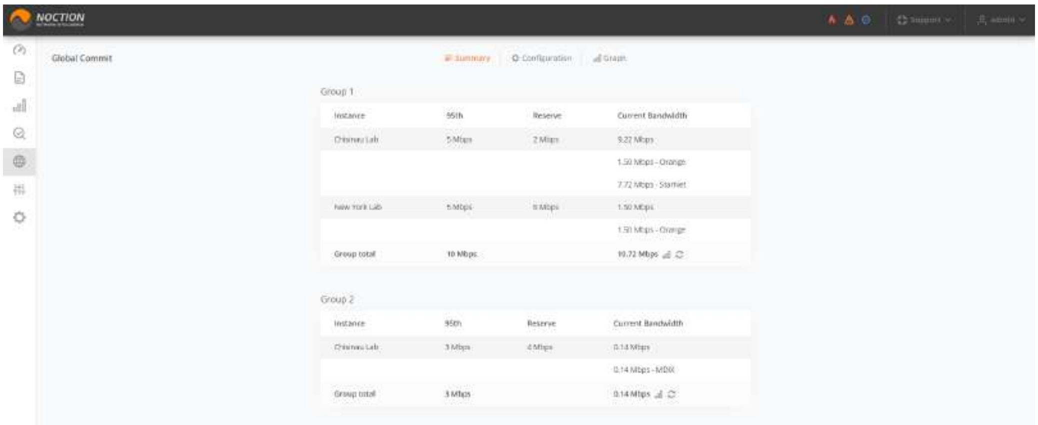

Figure 3.11.4: Global Commit summary

3.11.3 Global Commit Graph

This tab offers graphical representation of the bandwidth usage for a selected group, instances within the selected group as well as individual providers.

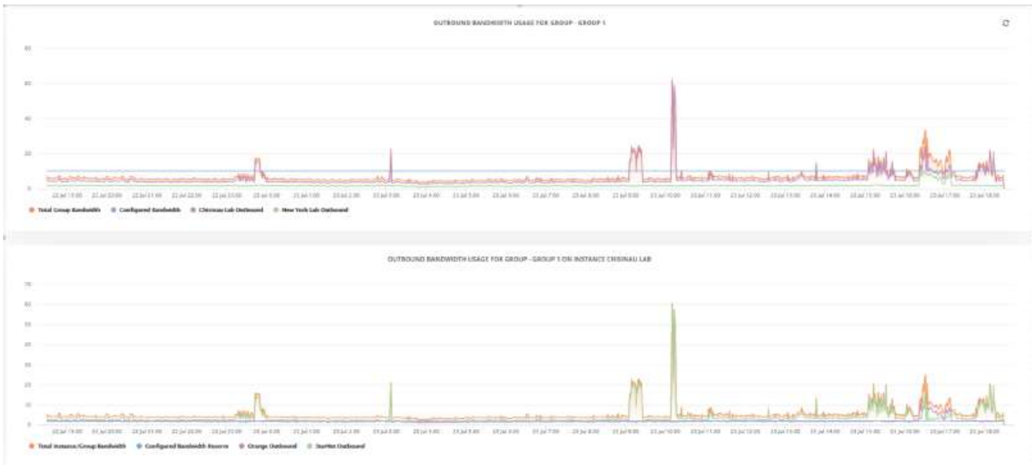

Figure 3.11.5: Global Commit Graphs

3.12 Configuration editor

The individual IRP instance system parameters can be modified from the Frontend, using the "Configuration" menu.

Figure 3.12.1: System events

When multiple GMI users are editing specific configuration settings for a given IRP instance, a notification shows up in the top right corner, highlighting their usernames. Such information helps to better coordinate the application of any changes to the configuration settings.

Figure 3.12.2: Concurrent editing of the Configuration Settings

3.12.1 Global configuration

Global settings can be adjusted under the Global tab.
Several global parameters can be configured by selecting their desired values and clicking on the "Save" button.
Available parameters:

- **Improvement mode** (global.improve_mode)
- **Management interface** (global.master_management_interface)
- **BGP mode** (Intrusive / Non-intrusive, see global.nonintrusive_bgp)
- **Probing interface(s)**. See global.master_probing_interface and others.

Click "Show advanced" to open a complete list of options. See section 4.2 for details on each parameter type.

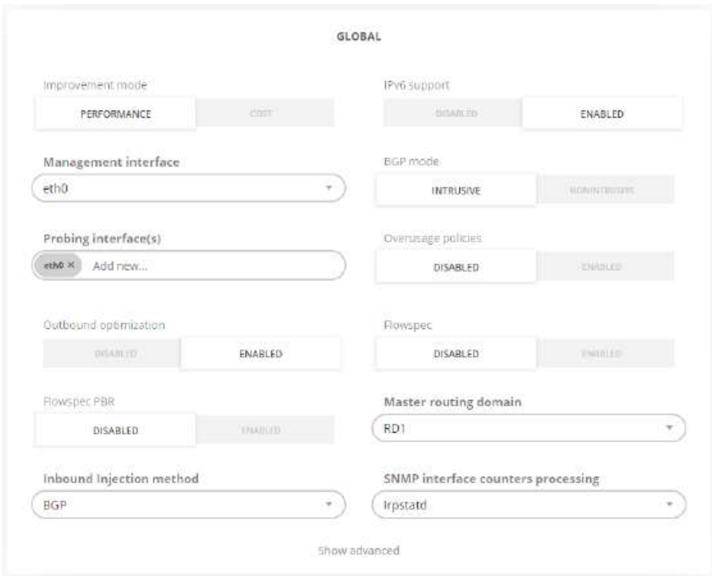

Figure 3.12.3: Configuration editor: Global settings

The list of ignored networks can be modified on the same page. The options allow listing as appropriate:

- prefixes(global.ignorednets),
- ASNs (global.ignored.asn),
- BGP Community attributes that mark prefixes/routes to ignore (global.ignored_communities).

Setting prefixes and ASNs is possible either manually, or by importing a text file containing one IP/network in CIDR format or ASN per line.

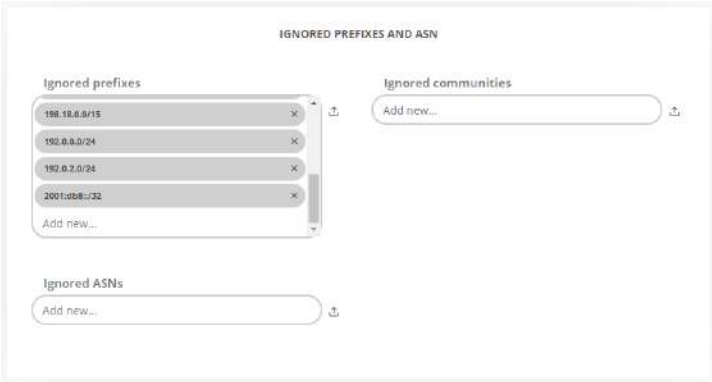

Figure 3.12.4: Configuration editor: Ignored prefixes

The Routing Domains can be configured from the Global tab as well.

Figure 3.12.5: Configuration editor: Routing Domains

3.12.2 BGP and Routers configuration

Specific IRP instance BGP daemon-related parameters can be configured under the "Configuration→BGP" tab.

- Bgpd parameters:
  - BGP mode intrusive/non-intrusive (global.nonintrusive_bgp)
  - BGP monitor enabled/disabled, this option is useful during DoS/DDoS attacks. Bgpd ICMP / SNMP monitors can be disabled on the fly, without restarting the Bgpd daemon.
  - AS-Path attribute a restore priority (bgpd.as_path)
  - Remove prefixes on next-hop update (bgpd.improvements.remove.next_hop_eq)
  - Remove prefixes on aggregate withdraw (bgpd.improvements.remove.withdrawn)
- BGP monitoring settings:
  - Guard time (bgpd.mon.guardtime)
  - Holdtime (bgpd.mon.holdtime)
  - Keepalive (bgpd.mon.keepalive)

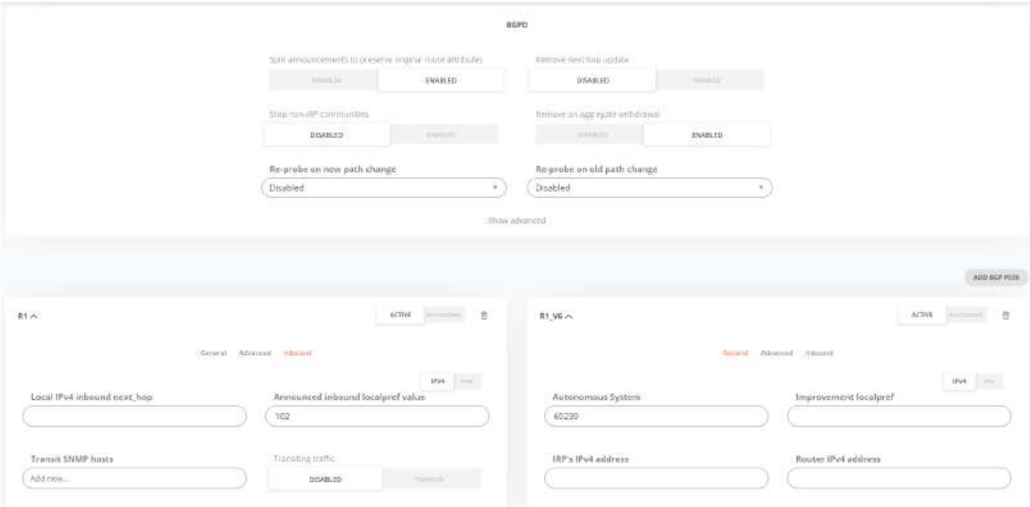

Figure 3.12.6: Configuration editor: BGP settings

BGP routers can be added, removed, activated, shutdown or modified on the same page.

A new BGP neighbor can be added using the "Add BGP Peer" button. Provide a name and configure the following parameters:

- General settings:
  - ASN to be used for the iBGP session (bgpd.peer.X.as)
  - Announced local-pref value (bgpd.peer.X.master_localpref)
  - Local IP address (bgpd.peer.X.master_our_ip, bgpd.peer.X.master_our_ipv6)
  - Neighbor IP address, usually the router's IP (bgpd.peer.X.master_peer_ip, bgpd.peer.X.master_peer_ipv6)

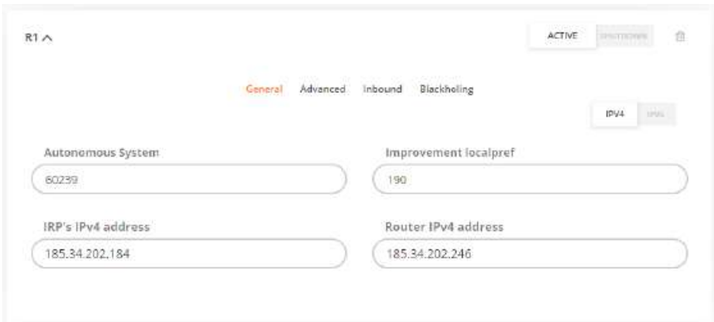

Figure 3.12.7: Configuration editor: New Router, general settings

- Advanced settings:
  - BGP session password
  - Improvement communities to be appended to the Communities attribute (bgpd.peer.X.master_communities)
  - Keepalive (bgpd.peer.X.keepalive)
  - Announced MED value (bgpd.peer.X.med)
  - BGP Router ID, mandatory for IPv6 only (bgpd.peer.X.slave_router_id)

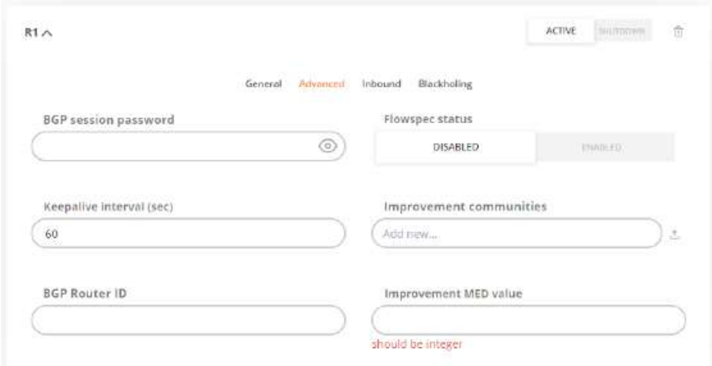

Figure 3.12.8: Configuration editor: New Router, advanced settings

- Inbound settings.
  - Local inbound next_hop (4.5.8)
  - Announced inbound LocalPref value (4.5.10)
  - Transit SNMP hosts (4.5.34)
  - Transiting traffic toggle (4.2.28)

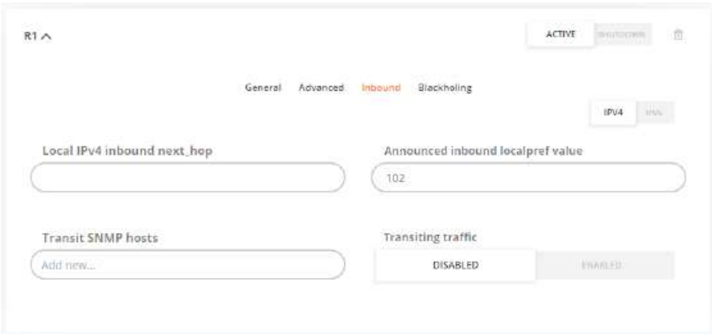

Figure 3.12.9: Configuration editor: New Router, Inbound settings

- Blackholing.
  - Announced blackholing localpref value
  - Blackholing IPv4 next hop
  - Blackholing IPv6 next hop

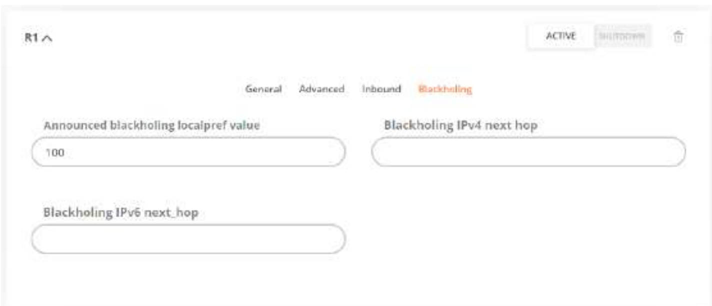

Figure 3.12.10: Blackholing configuration settings

3.12.3 Collector configuration

Global collector parameters, as well as Span and Flow-specific settings can be adjusted under "Configuration→Collector" tab.

- Global collector settings:
  - Top volume prefixes per cycle (collector.export.volume.high.top_n)
  - Minimal traffic volume (collector.export.volume.min)
  - Minimal traffic volume (%) (collector.export.volume.min_pct)
- Flow collector settings:
  - UDP port to listen for NetFlow or sFlow packets (collector.flow.listen.nf, collector.flow.listen.sf)
  - Flow sources (collector.flow.sources)
- SPAN collector settings:
  - Capture interfaces (collector.span.interfaces)
  - Mindelay algorithm enable/disable (collector.span.min_delay)
  - Mindelay probing queue slots (collector.span.min_delay.probing_queue_size)

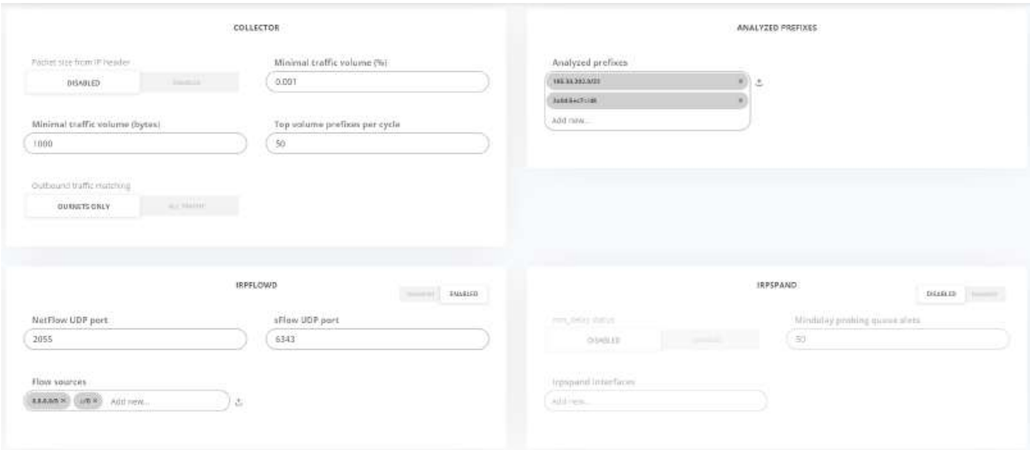

Figure 3.12.11: Configuration editor: Collector settings

The list of prefixes announced by the monitored network can be edited in the same form. Its contents can be also imported from a text file.
See also: collector.ournets.

- Inbound Prefixes collector settings:
  - prefix belonging to the network (4.15.6),
  - router(s) where announcements are sent (4.15.1),
  - next hop used by improvements for this prefix (4.15.5)
  - an option to instruct IRP whether to fully control the prefix or to only announce improvements when there are any.
  - providers for which this inbound prefix can be optimized (4.15.7). It must be noted that if no provider is selected then the prefix is optimized for all providers. Otherwise, the prefix will be optimized only through selected providers.
  - prefix status as each prefix can be disabled in order to exclude from further inbound optimization (4.15.3)

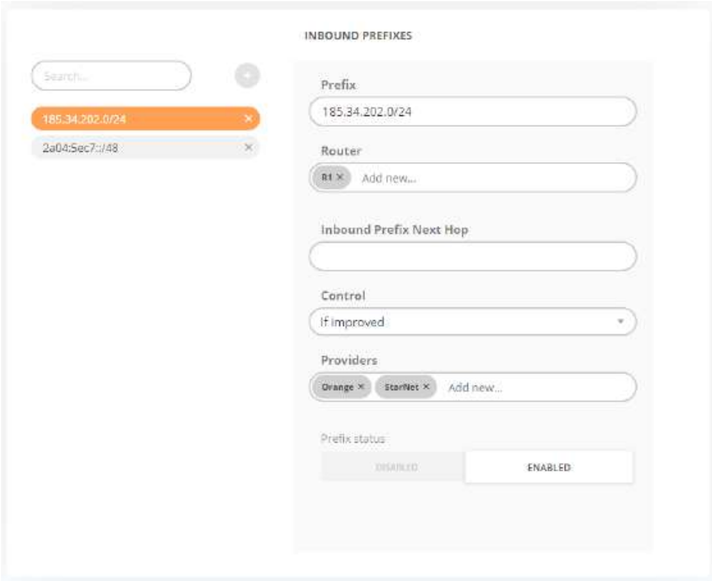

Figure 3.12.12: Configuration editor: Collector Inbound Prefixes settings

3.12.4 Core configuration

All Core parameters affecting the decision-making algorithms can be modified on the next tab ("Configuration→Core").

The following configuration parameters can be adjusted:

- **Max IPv4 improvements** (core.improvements.max, core.improvements.max_ipv6)
- **Standard reprobing period** (core.improvements.ttl.retry_probe) - the time period after which a specific improvement is being scheduled for re-probing.
- **VIP reprobing period** (core.vip.interval.probe) - defines the VIP prefixes probing interval, in seconds.
- **Minimal probe lifetime** (core.probes.ttl.min) - Ordinary probing will not be performed for a specific prefix if its probe age is lower than this value.
- **Allowed latency worsening** (core.cost.worst_ms)
- **Exploring queue slots** (core.eventqueuelimit)
- **Relevant loss** for loss-based improvements (core.performance.loss_pct) - a prefix will be improved based on a loss cause only if the packet loss can be decreased by this value (in %)
- **Relevant RTT difference** (core.performance.rtt.diff_ms)
- **Relevant RTT difference** (%) (core.performance.rtt.diff_pct)
- **Maximum probe lifetime** (core.probes.ttl.max)

Figure 3.12.13: Configuration editor: Core configuration

Outage detection algorithm can be enabled and configured on the same page. Outage detection algorithm can be enabled or disabled using the toggle buttons at the top right.

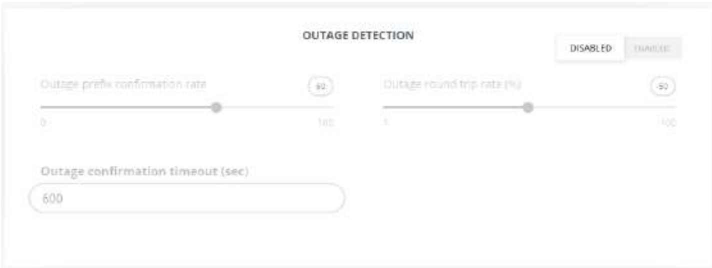

Figure 3.12.14: Configuration editor: Outage detection

Throttling with Flowspec can be configured on the same page.

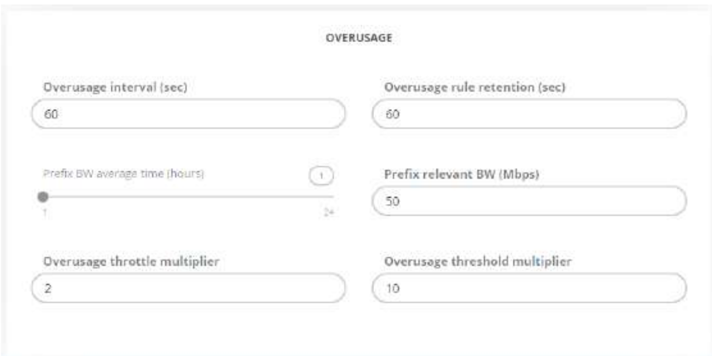

Figure 3.12.15: Configuration editor: Overusage

The following configuration parameters can be adjusted:

- **Overusage interval** (core.overusage.check_interval) sets the frequency in seconds of checking for excessive bandwidth use.
- **Overusage rule retention** (core.overusage.hold_timer) sets how much time a rule is kept after bandwidth use returns to normal.
- **Prefix BW average time** (core.overusage.out.average.period) sets the number of hours used to determine the average bandwidth use of a prefix.

- **Prefix relevant BW** (core.overusage.out.average.relevant_min) sets the relevant bandwidth use in Mbps by a prefix before considering any rules for it.
- **Overusage throttle multiplier** (core.overusage.out.threshold.throttle) sets the multiplier to apply to average prefix bandwidth use when setting a rule.
- **Overusage threshold multiplier** (core.overusage.out.threshold.trigger) sets the threshold multiplier used to determine excessive bandwidth use.

Prefix relevant BW and Overusage threshold multiplier parameters control the number of Flowspec rules that will be generated automatically. Adjust these values to control the number of Throttling Flowspec policies.

Circuit Issues Detection parameters are listed and can be adjusted:

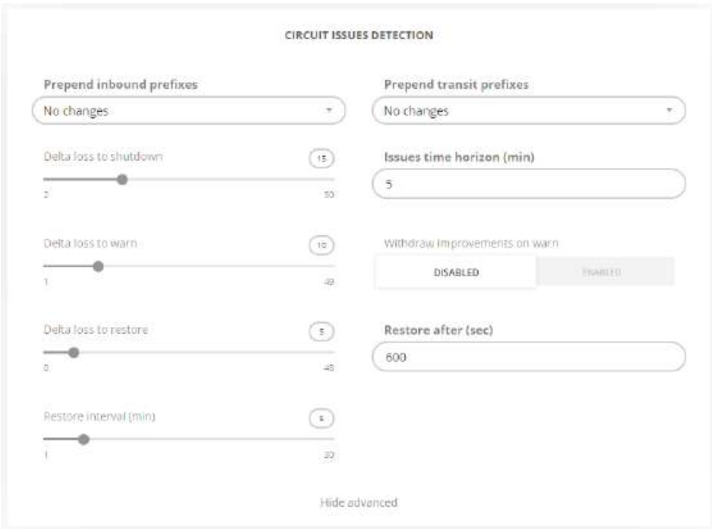

Figure 3.12.16: Configuration editor: Circuit issues detection

The following configuration parameters can be adjusted:

- **Delta loss to shutdown** (core.circuit.high_loss_diff) sets threshold to initiate shutting down a provider with circuit issues as a difference between examined provider's average loss and overall average loss during the given time horizon. Shutdown is attempted only for providers marked accordingly.
- **Delta loss to warn** (core.circuit.warn_loss_diff) sets threshold to raise a warning regarding issues with a provider as difference between examined provider's average loss and overall average loss during the given time horizon. Usually this threshold is significantly lower than the shutdown threshold.
- **Delta loss to restore** (core.circuit.recover_loss_diff) sets low loss level threshold when IRP will restore full functionality over provider that had circuit issues.
- **Issues time horizon** (core.circuit.hist_interval) sets how many minutes in the past IRP looks for probes with loss over both analyzed provider and all the other providers on the network.
- **Withdraw improvements on warn** (core.circuit.withdraw_on_warn) instructs IRP to withdraw outbound improvements over provider with circuit issues when warning threshold is reached. By default improvements are withdrawn only when shutdown threshold is exceeded.
- **Restore after** (core.circuit.recover_hold_time) sets the interval in seconds after which IRP should re-evaluate a provider's circuit loss and attempt restoring it to normal function. This will be performed only for providers that are configured to attempt to restore after shutdown.

- **Restore interval** (core.circuit.recover_monitored_intervals) sets the interval in minutes during which IRP will periodically re-evaluate a provider's circuit loss and attempt restoring it to normal function. This will be performed only for providers that are configured to attempt to restore after shutdown.

The configuration parameters above are applied for all providers even though for individual providers different level of control can be set starting from disabling and ending with attempting both to automatically shutdown and later restore connectivity with it. See also Providers configuration

3.12.5 Commit Control configuration

Commit Control algorithm can be enabled and configured on the Commit Control group of pages. It can be enabled or disabled using the toggle On/Off buttons at the top of the form.

If NetFlow is used to collect statistics Routers MUST be configured to export statistics every minute (or as often as possible). Some router models have default export intervals for either inactive or active flows of up to 1800 seconds. Big delays cause IRP to react very slowly to increased load and reduce the effectiveness of Commit Control feature.

The most important parameters for Commit Control are, as follows:

- **Commit Control probing queue slots** (core.commit_control.probing_queue_size)
- **Minimal prefix bandwidth** in Mbps (core.commit_control.agg_bw_min)

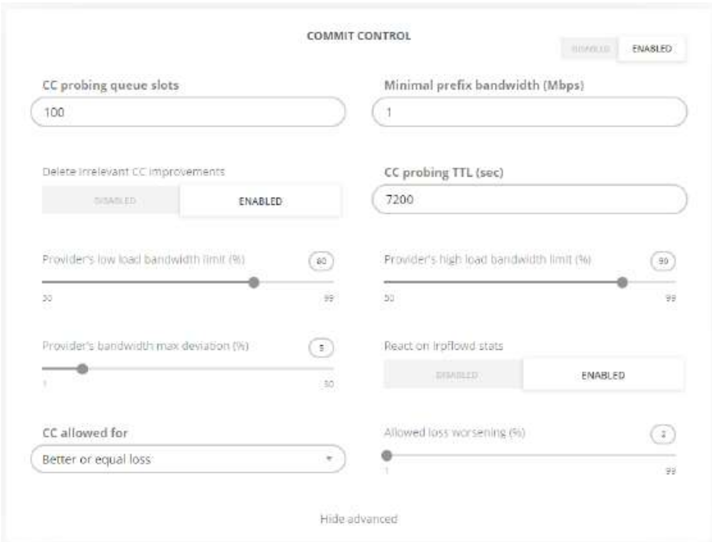

Figure 3.12.17: Configuration editor: Commit Control

Providers Overall block displays Commit Control configuration for all providers including their precedence. See details in the figure below.

Individual provider parameters can be adjusted as well as detail regarding current Commit Control changes applied by an IRP instance can be accessed by following the provided links.

PROVIDERS OVERALL

Provider	Status	95th Mode	Configured 95th for Inbound	Group Configured 95th for Inbound	Configured 95th	Group Configured 95th	Current BW Usage	Monthly 95th	Cost	Load Balancing	Precedence
Orange	CC Enabled	95th from greater of in, out	3	-	5	-	0.00 Mbps	0.00 Mbps	2	Enabled	2
StarNet	CC Enabled	95th from greater of in, out	6	-	10	-	0.00 Mbps	0.00 Mbps	4	Enabled	3
MDIX	CC Enabled	95th from greater of in, out	-	-	100	-	0.00 Mbps	0.00 Mbps	-	Enabled	10

Figure 3.12.18: Providers Overall

3.12.5.1 Inbound Commit Control

Inbound bandwidth control is part of the Commit Control feature. You need to enable Commit Control and further to enable Inbound Commit control to make use of this feature.

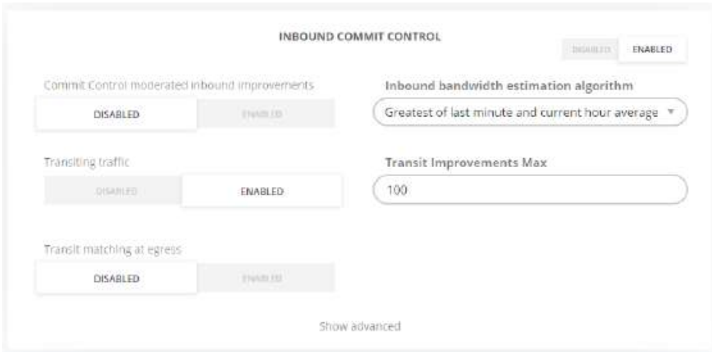

Figure 3.12.19: Configuration editor: Inbound commit control

Commit control for inbound highlights the relevant configuration parameters:

- whether inbound commit control is on or off
- whether review and moderation feature is enabled or disabled
- what is the bandwidth estimation algorithm for inbound and others

Refer also to peer.X.95th.mode, core.commit_control.inbound.enabled, core.commit_control.inbound.moderated.

Besides the above mentioned settings for Inbound Commit Control separate low and high limits are configured to instruct IRP when to start and when to stop improving and also how to estimate the effects of inbound improvements.

Estimating inbound traffic Internet traffic has high variability and adjacent measurements can differ significantly between them. In these conditions a leveling function helps mitigate the risk of generating excessive numbers of Inbound improvements due to higher traffic variability of some networks. This estimation can be fine tuned by the Inbound bandwidth estimation algorithm parameter that tells an IRP instance what value to use as the basis of the calculation. See the possible base values below:

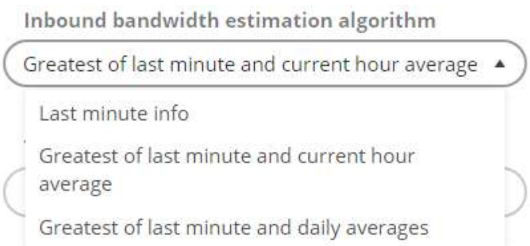

Figure 3.12.20: Configuration editor: Inbound BW estimation algorithms

3.12.5.2 Optimization of transiting traffic

Optimization of transiting traffic is setup as part of Inbound Commit Control.

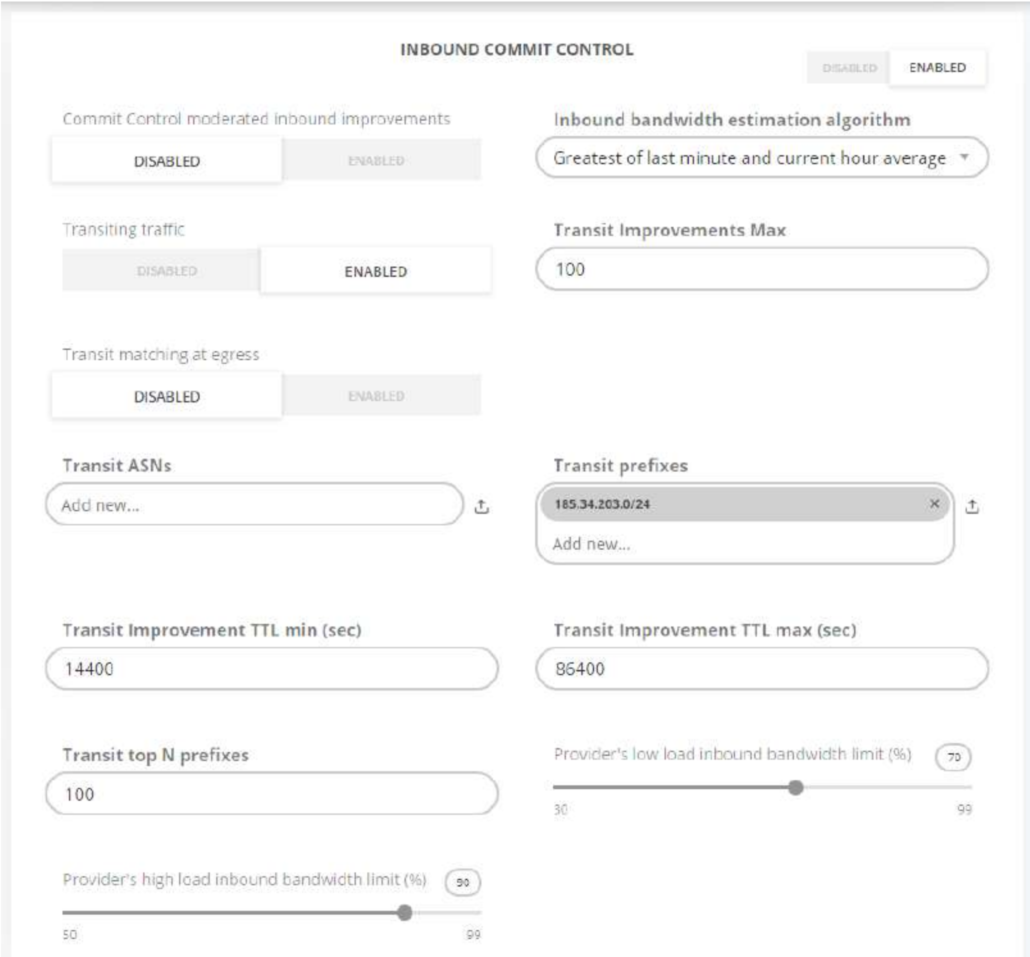

Figure 3.12.21: Configuration editor: Inbound optimization of transiting traffic parameters

The following configuration parameters can be adjusted:

- Transiting traffic toggle that enables or disables the feature (global.inbound_transit)
- Transit Improvements Max sets the maximum number of transit improvements (core.improvements.inbound_transit.max)
- Transit ASNs and Transit prefixes specify those segments of the Internet that IRP monitors and optimizes (bgpd.prefixlist.asn, bgpd.prefixlist.prefixes)
- Transit Improvements TTL min and max set the lower and upper bounds in seconds to keep a transit improvement (core.improvements.inbound_transit.ttl.min, core.improvements.inbound_transit.ttl.max)

Note that traffic belonging to a specific prefix that fragments a large transit prefix is collected as belonging only to the specific prefix and excluded from the larger prefix traffic statistics. If the specific prefix is also filtered from IRP configuration as for example is the case of a /26 prefix, then the specific prefix will be excluded from optimization of transiting traffic and will not show up in any of the subsequent decisions or reports.

- Transit Top N prefixes sets the number of transit prefixes that are collected each cycle and considered for optimization (collector.flow.export.inbound_transit.topn)

- Match transit at egress enables or disable statistics collection for transit prefixes when packets exist the network (collector.flow.process_transit_in_outbound)

3.12.6 Inbound performance optimization configuration

Refer to Inbound performance optimization for the Inbound Performance feature overview.

To adjust the Inbound performance optimization parameters, go to “Configuration→Inbound”.

The following configuration parameters can be adjusted:

- **Inbound performance (irpinperfd.enabled)** enables/disables inbound performance optimization. This also disables the traffic statistics collection, required for inbound performance optimization
- **Inbound Performance moderated mode (irpinperfd.moderated)** enables/disables manual moderation of all decisions of the inbound performance optimization algorithm
- **Probe shelf life (irpinperfd.probing.shelf_life)** sets the maximum age of a probe which can be reused
- **Probing failure margin (irpinperfd.probing.failure_margin)** sets the percentage of probing failures (maximum possible value is 25%), exceeding which is critical and leads to the inbound performance optimization algorithm restart
- **Probing timeout (irpinperfd.probing.timeout)** sets the time period of probing process, exceeding which is critical and leads the inbound performance optimization algorithm restart
- **Reprobing interval (irpinperfd.probing.interval)** sets the period of time to wait until the algorithm starts reprobing a sample and comparing it against the model
- **Top volume prefixes per inperf rule (irpinperfd.model.topn_per_rule)** sets the number of prefixes to be considered for the rule’s model construction. Prefixes with greater volume are taken into account first

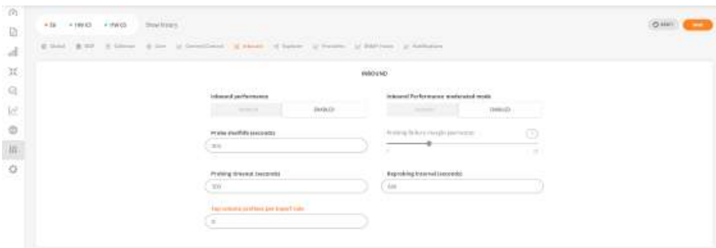

Figure 3.12.22: Configuration editor: Inbound configuration settings

The Inbound Performance feature makes it’s decisions based on the configured rules.

When the Moderated mode is enabled, the Inbound Performance improvements should be applied manually at the Inbound Performance Rules page.

Refer to Inbound Performance Rules, Inbound Performance Optimization and Inbound Performance .

3.12.7 Explorer configuration

To adjust the Explorer-related parameters, go to “Configuration→Explorer”. The Explorer settings sections should be consulted for detailed Explorer parameters description.

Explorer parameters:

- Infrastructure IPs (explorer.infra_ips)
- Explorer worker threads (explorer.maxthreads)
- Probing algorithms and Traceroute algorithms (explorer.probe.algorithm, explorer.trace.algorithms)
- High volume task precedence (explorer.high_vol_precedence)
- Process max collected IPs (explorer.max_collector_ips)
- First probing packets amount (explorer.probing.sendpkts.min)
- Adaptive probing packets count (explorer.probing.sendpkts.adaptive_max)
- ICMP timeout (explorer.timeout)
- Traceroute retry packets and packets per hop (explorer.traceroute.sendpkts, explorer.traceroute.retrypkts)
- Traceroute minimum and maximum ttl (explorer.traceroute.ttl.min, explorer.traceroute.ttl.max)

Figure 3.12.23: Configuration editor: Explorer settings

3.12.8 Providers and Peers configuration

Providers can be added and modified via “Configuration→Providers”.
See also: Provider

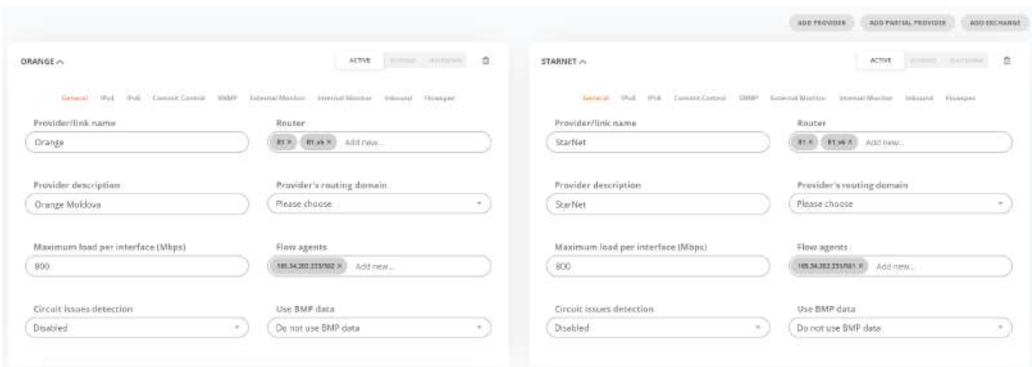

Figure 3.12.24: Configuration editor: Providers configuration

The provider of choice widget should be expanded to be able to edit it. Using the control buttons, a specific provider can be temporarily suspended (e.g for a short maintenance in the monitored network), shutdown (for long maintenance), or completely deleted. See also: peer.X.shutdown. Commit Control can also be disabled or enabled for each provider specifically. (peer.X.cc_disable).

The sections below describe the most common groups of provider configuration parameters. Some parameters are repeated on more than one of these groups for convenience.

3.12.8.1 Providers configuration

To add a new provider, the "Add provider" button must be used. Several parameters should be configured for the new provider:

- General settings:
  - Provider name (peer.X.shortname)
  - Router (peer.X.bgp_peer)
  - Provider description (peer.X.description)
  - Provider's routing domain (peer.X.rd)
  - Maximum load per interface (peer.X.limit_load)
  - Flow agents (peer.X.flow_agents)
  - Circuit issues detection (peer.X.circuit.control) indicating how IRP should react to excessive persistent loss over a particular provider.
  - Use of BMP data (4.6)

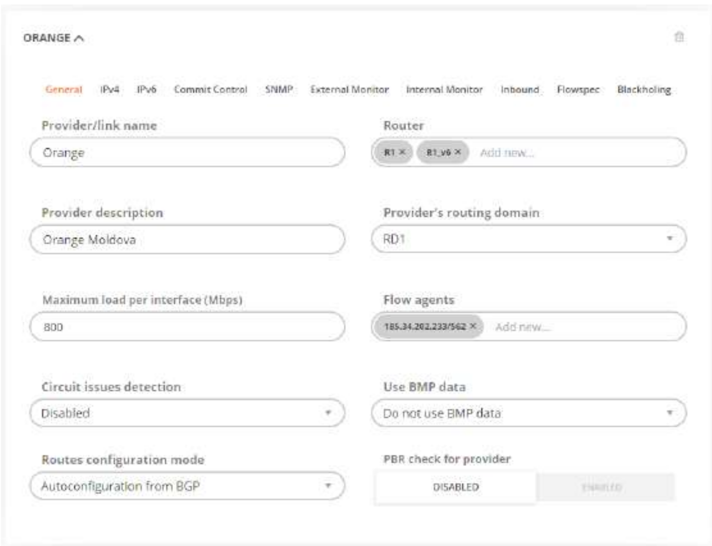

Figure 3.12.25: Configuration editor: Adding a new provider, General settings

- IPv4 / IPv6 settings:
  - IPv4 / IPv6 diagnostic hops (peer.X.ipv4.diag_hop, peer.X.ipv6.diag_hop)
  - Probing IPv4 / IPv6 address (peer.X.ipv4.master_probing, peer.X.ipv6.master_probing)
  - Remote provider ASN (peer.X.ipv4.next_hop_as), used by the AS-Path restoration algorithm (bgpd.as_path)
  - Router next-hop address (peer.X.ipv4.next_hop), that sets the next-hop value for injected routes related to this provider
  - Provider's alternative IPv4 address for PBR tests (4.14.35)

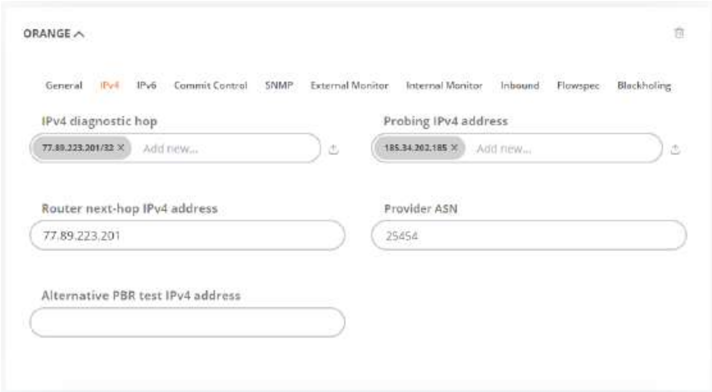

Figure 3.12.26: Configuration editor: Adding a new provider, IPv4 / IPv6 settings

- Commit Control configuration:
  - Provider cost per Mbps (peer.X.cost)
  - Provider 95th percentile (peer.X.95th)
  - Provider inbound 95th percentile (peer.X.95th.in)
  - Provider 95th calculation mode for inbound traffic
  - Provider billing day (peer.X.95th.bill_day)
  - Commit Control status for this provider (peer.X.cc_disable)
  - CC provider precedence - used for Commit Control and grouping (peer.X.precedence)
  - Centile value (peer.X.95th.centile)
  - Performance/Cost improvements within provider group toggle in case the peer load balancing is configured, (peer.X.improve_in_group, peer.X.precedence)

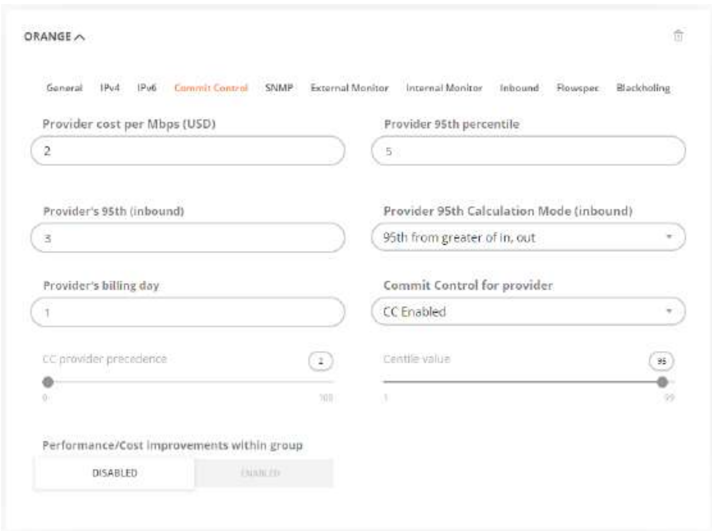

Figure 3.12.27: Configuration editor: Adding a new provider, Commit Control

- SNMP-related settings:

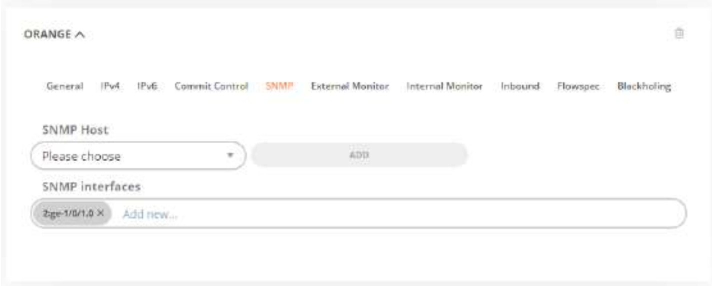

Figure 3.12.28: Configuration editor: Adding a new provider, SNMP settings

- External Monitor settings:
  - External monitor status toggles for IPv4 / IPv6 (peer.X.mon.ipv4.external.state, peer.X.mon.ipv6.external.state)
  - ICMP/UDP ping monitored IPv4 / IPv6 addresses (peer.X.ipv4.mon, peer.X.ipv6.mon)

Figure 3.12.29: Configuration editor: Adding a new provider, External Monitor

- Internal Monitor settings:
  - Internal monitor status toggles for IPv4 / IPv6 (peer.X.mon.ipv4.internal.state, peer.X.mon.ipv6.internal.state)
  - BGP session monitoring IPv4 / IPv6 addresses (peer.X.mon.ipv4.bgp_peer, peer.X.mon.ipv6.bgp_peer)
  - BGP MIBs for IPv4 / IPv6 (peer.X.mon.ipv4.internal.mode, peer.X.mon.ipv6.internal.mode)
  - SNMP host for BGP Internal Monitor (peer.X.mon.snmp)

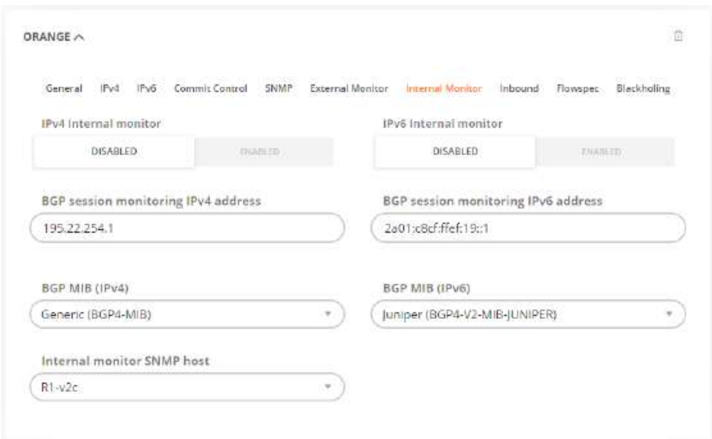

Figure 3.12.30: Configuration editor: Adding a new provider, Internal Monitor

- SNMP v3 uses additional parameters depending on security services used for monitoring:

- Inbound:
  - Base community for inbound improvements (4.14.27)

  Inbound optimization improvements advertise larger or smaller counts of prepends to be announced to different providers. IRP instances use BGP session to communicate to routers and relies on BGP communities to communicate the improvements. Each provider is assigned a base community which represents zero prepends. Additional prepends are represented by incrementing accordingly the right-side part of the provider's community.

⚠ IRP instances use 8 prepend levels. This means that base community should be chosen to allow additional 7 values without intersecting with other community values.

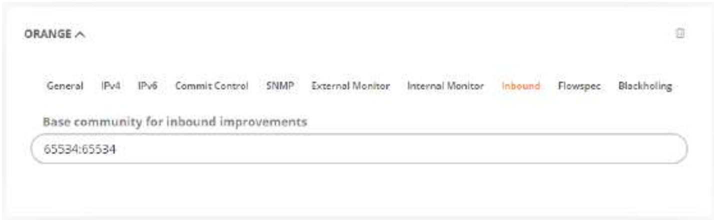

Figure 3.12.31: Configuration editor: Adding a new provider, Inbound

- Flowspec:
  - IPv4 / IPv6 Redirect community (4.14.20 and 4.14.21)

Figure 3.12.32: Configuration editor: Adding a new provider, Flowspec

- Blackholing:
  - Blackholing Routers
  - Blackholing community

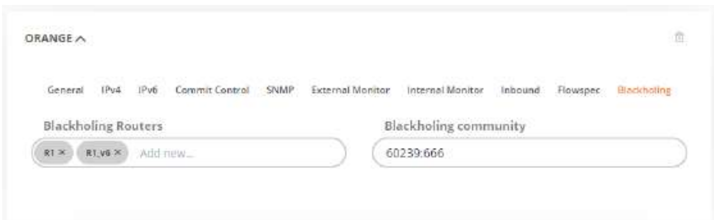

Figure 3.12.33: Provider Blackholing configuration details

The same parameters can be adjusted for the already configured providers.

3.12.8.2 Internet Exchanges configuration

An Exchange is configured as a special type of provider. The important things to consider when setting up an Exchange are listed below.

It is recommended that Noction systems engineers guide you through Exchange configuration process.

IRP needs one ACL and one PBR rule for each Peering Partner on an Exchange. Internet Exchanges can have hundreds of Peering Partners. Ensure that the number of Access Lists, Access List entries and PBR entries will not exceed router hardware limitations.

- Once the Exchange is setup via GMI, an IRP instance will retrieve the routing table on the router in order to setup Exchange peering partners. IRP will require access to the router in order to do so.

When configuring an Exchange its Diag Hop parameter must be provided. Diag Hop for an Exchange represents the list of direct-connected networks configured on the router's interface towards the Exchange network. This parameter is labeled "IPv4 diagnostic hop" on IRP Frontend.

- Probing IPs for Exchanges no longer require one IP address per peering partner since this number might be impractically big. Instead a combination of probing IP and DSCP value is used to configure the PBRs. A single probing IP in conjunction with DSCP can cover up to 64 peering partners. A sufficient number of probing IPs will be required in order to cover all peering partners on your exchange.

As a rule of thumb it is better to list the PBR rules for transit providers before the (large number of) rules for Internet Exchange peers. If the many PBR rules assigned to peers on a large Internet Exchange are placed at the beginning of the PBR list some routers evaluate all of them before finding the terms for transit providers and this consumes valuable router CPU resources.

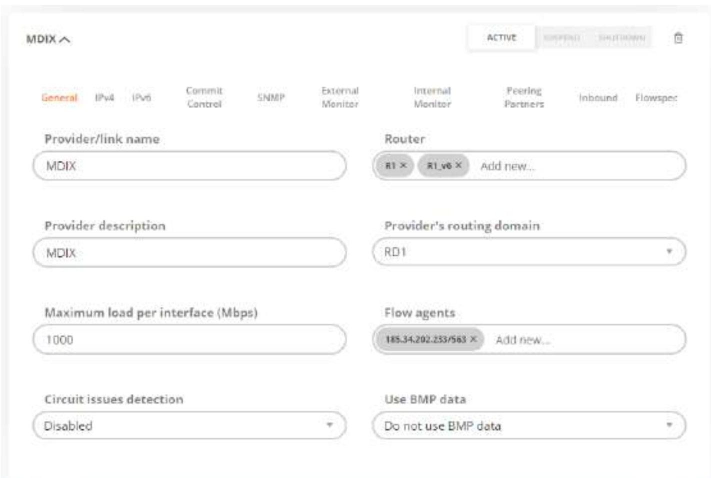

Figure 3.12.34: An Internet Exchange is similar to a provider and it requires configuration of the Router, the Probing IPs and the other usual attributes

- Once the Exchange is configured, the IRP instance will need to reset its BGP session towards the router interconnecting with the Exchange in order to retrieve the required routing table information about all peering partners. Once the BGP session is restarted IRP instance will start fetching Exchange routing tables.

Allow sufficient time (~10 minutes) to fetch Exchange routing tables before continuing configuration.

- Peering partner autoconfiguration function is available. It retrieves the list of Next Hops (peering partners) on the Exchange with the corresponding list of prefixes individual peers accept.

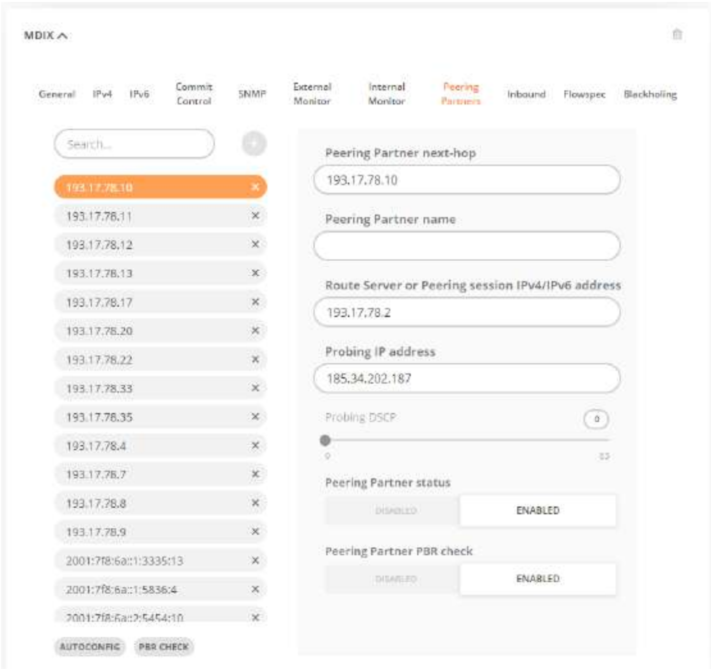

Figure 3.12.35: The list of peering partners for the Exchange shows details about each of them and offers access to Autoconfiguration feature and PBR ruleset generator for the router

- Use the Autoconfiguration feature to create an initial configuration for an IRP instance. Review Exchange peering partners before starting the Exchange. Consider enabling periodic auto re-configurations for selected IX in order to update periodically the value of BGP session monitoring IPv4 address from the Exchange and to add new peering partners.
  Refer global.exchanges.auto_config_interval and peer.X.auto_config.

Keep in mind that Autoconfiguration might tamper with all the changes you've made directly within IRP instance config files for peering partners. So, use Autoconfiguration sparingly after you've applied manual changes or try avoiding direct configuration file changes altogether.

- Once the Exchange is configured correctly, you will need to apply the PBR rules on the router(s). There is a functionality to generate PBR rules for different router vendors. You will need to review the generated ruleset and apply it on the router manually.
- It is possible that the Autoconfiguration feature on the Exchange has been run with incomplete routing tables or that new peering partners have been connected. This is especially true for very big Exchanges. In this case it is possible that some peering partners are neither in IRP nor router configurations. When such a case is detected, a warning about newly discovered peering partners gets raised. At this stage you will need to both re-autoconfigure IRP instance and extract the PBRs for the router.

- After its creation and up to its complete configuration the Exchange is kept in a Stopped state. When both IRP and the Router PBRs are configured, particular IRP instance is ready to start optimizing Exchange traffic too. Keep in mind that starting the Exchange will require a BGP session restart.

> We must mention that before applying changes to IRP instance configuration the changes are validated. If errors are detected these will be flagged and the previous good configuration will be kept intact so you will have the option to review the erroneous data and correct.

3.12.8.3 Switching a provider from one router to another

Switching a provider from one router to another is possible via manual IRP instance reconfiguration only.

Use the following steps to switch the provider from one router to another:

1. Suspend the provider using GMI Frontend "Providers and Peers" particular IRP configuration section. IRP instance withdraws all existing and stops new improvements towards shutdown provider(s)
2. You can remove traffic from the provider (by denying incoming/outgoing announces in the BGP configuration) and then physically re-cable the provider from one router to another. Then configure BGP session towards the provider on new router
3. The PBR rule for the provider should be configured on new router (move provider's PBR settings from the old router to the new one)
4. Change IRP box local PBR rules if any
5. Check if PBR works properly using traceroute tool
6. Modify (/etc/noction/irp.conf) assigned router for the provider (refer to peer.X.bgp_peer)
7. Modify (/etc/noction/irp.conf) SNMP interface/IP/community for the provider (refer to SNMP Host, peer.X.snmp.interfaces, peer.X.mon.snmp)
8. Reload Bgpd configuration (affected BGP sessions could be reset)
9. Re-activate the provider using GMI Frontend "Providers and Peers" IRP instance of choice configuration section
10. Check IRP BGP Internal and External monitor states. They must be UP.

3.12.9 SNMP hosts configuration

SNMP hosts are nodes on the network that provide or read SNMP data. Multiple SNMP hosts are configured in this section of configuration and references to them are used from elsewhere in the configuration. SNMP hosts can be configured by navigating via to "Configuration→SNMP hosts" tab.

See also: SNMP Host

Figure 3.12.36: Configuration editor: SNMP hosts configuration

- SNMP v3 uses additional parameters depending on security services used for statistics collection:
  - SNMP security services selects if Authentication and/or Privacy services are used (snmp.X.seclevel)
  - SNMP authentication password if authentication is used (snmp.X.auth_password)
  - SNMP authentication protocol if authentication is used (snmp.X.auth_protocol)
  - SNMP encryption password if privacy is used (snmp.X.priv_password)
  - SNMP encryption protocol if privacy is used (snmp.X.priv_protocol)
  - SNMP Username if authentication is used (snmp.X.auth_username)
  - SNMP version (snmp.X.version)

3.12.10 IRP instance Notifications

Notifications are messages sent to alert about some events registered by IRP instances. There are two prerequisites in order to start using notifications:

1. Configure event parameters and channels
2. Subscribe for event notifications

IRP instances can send notifications as SMS, email and SNMP Trap.

3.12.10.1 Configure notifications and events

Events configuration changes default threshold values when an event is raised. The most relevant events are presented in the following figure. For example Overloaded by (%) indicates that this event will fire when the aggregated outbound bandwidth usage for all providers exceeds by 10% the configured limits. Refer section 4.12 for details.

Figure 3.12.37: Configuration editor: Events configuration

IRP uses a local email server to send emails containing info on particular events registered by IRP instances. Still, emails sent by this server might trigger filtering policies enforced by some third parties that might block important event notifications. It is possible to provide the details of another email server on your network that is protected from this. The following figure highlights the available email channel configuration parameters. Besides specifying the server, email channel configuration sets the sender of email messages that will show in the receiver's inbox.

Only SMTP servers without authentication are currently supported for sending events registered by IRP instances. Set up a separate "sender" to receive specific reports and graphs via email. For details, refer to section 3.13.5

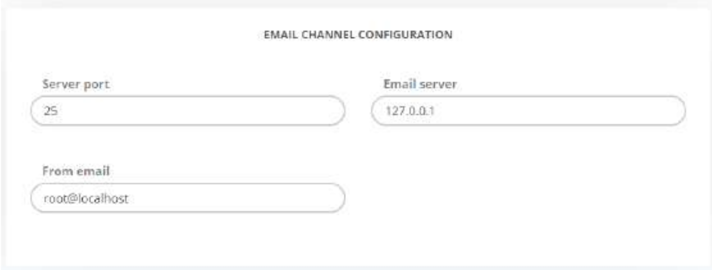

Figure 3.12.38: Configuration editor: Email configuration

SMS configuration is mandatory for sending SMS notifications.
The following SMS gateways are supported:

- Twilio
- Plivo

A valid account with a supported SMS gateway is required in order to finish configuration. Check the following figure for details:

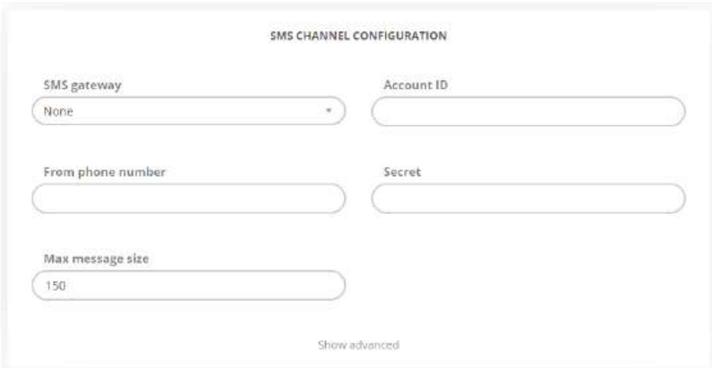

Figure 3.12.39: Configuration editor: SMS configuration

The settings are:

- SMS gateway - choose one of the supported gateways
- Account ID - the public identifier assigned to your account by the SMS gateway. The following figure highlights this value for Plivo
- From phone number - the phone number that shows as sender on the receiver's mobile device. Use a phone number supported by the SMS gateway
- Secret - the secret identifier assigned to your account by the SMS gateway. The following figure highlights this value for Plivo
- Max message size - the maximum length of the SMS text. The SMS will be trimmed before sending. Subsequently the SMS gateway will split the SMS into multiple parts if required.

Refer section 4.12 for complete details about configuration parameters.

⚠ Some SMS gateways enforce the From phone number and will reject numbers that do not comply with their policy.

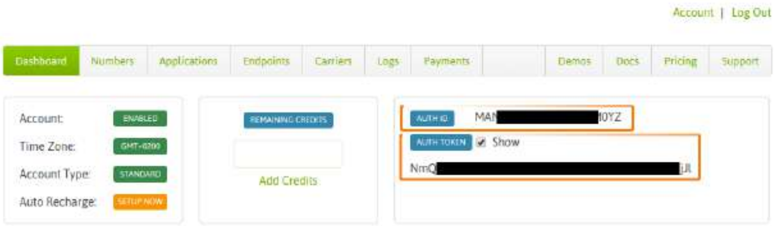

Figure 3.12.40: Plivo account ID and secret

There is a list of parameters that should be configured to deliver SNMP traps. Refer section 4.12 for details.

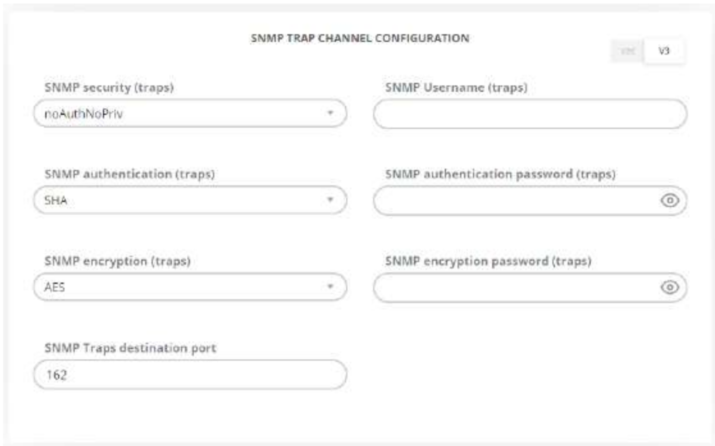

Figure 3.12.41: Configuration editor: SNMP Traps configuration

Web Hooks configuration requires only a Webhook URL to be provided. There is a series of advanced configuration parameters in case their default settings are not satisfactory. Refer section 4.12 for details.

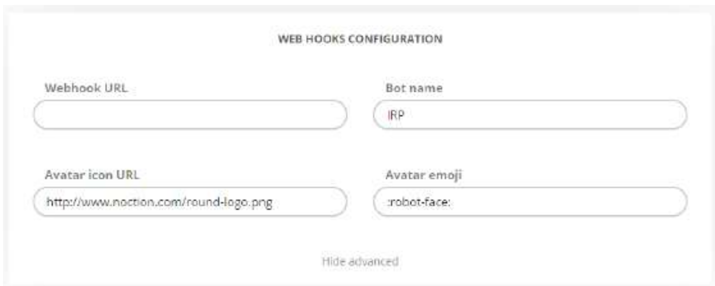

Figure 3.12.42: Configuration editor: Web Hooks configuration

The settings are:

- Webhook URL - the unique URL the team is assigned by Web hook provider. An example for Slack.com is shown
- Bot name - optional bot name assigned to Webhook bot
- Avatar icon URL - optional parameter that designates an icon (usually PNG or JPEG image are supported) assigned to the Webhook bot in the channel
- Avatar emoji - an alternative avatar specified in the form of an emoji in ":robot-face:" format.

⚠ Web hooks support has been tested and verified to work for Slack.com API.

3.13 Management

3.13.1 Instances

The “Instances” tab under “Management” accumulates information about all IRP instances added to GMI. The section presents IRP shortnames, status, hosts, license, IRP versions as well as the “Last Edited” and “Added by” details.

Admins are free to add new, edit, and delete IRP instances as they like.

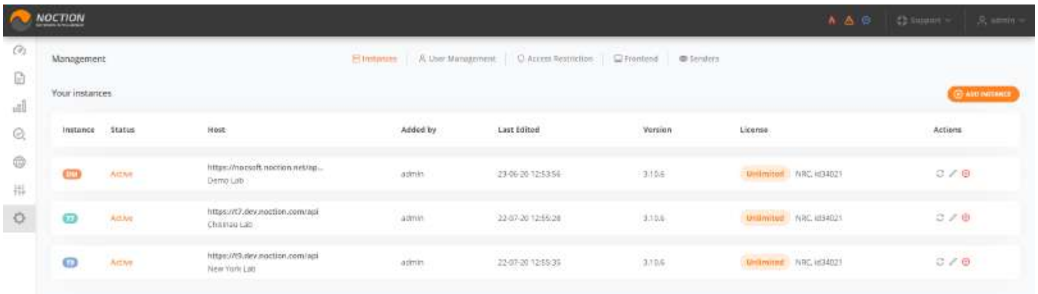

Figure 3.13.1: IRP Instances

To add a new instance, one should click the ADD INSTANCE button and fill out the corresponding fields in a popup window. The actual IRP instance authorization token or server root password should be provided for completing the registration.

For complete process breakdown see Registering IRP instances using tokens and Registering IRP instances using root passwords

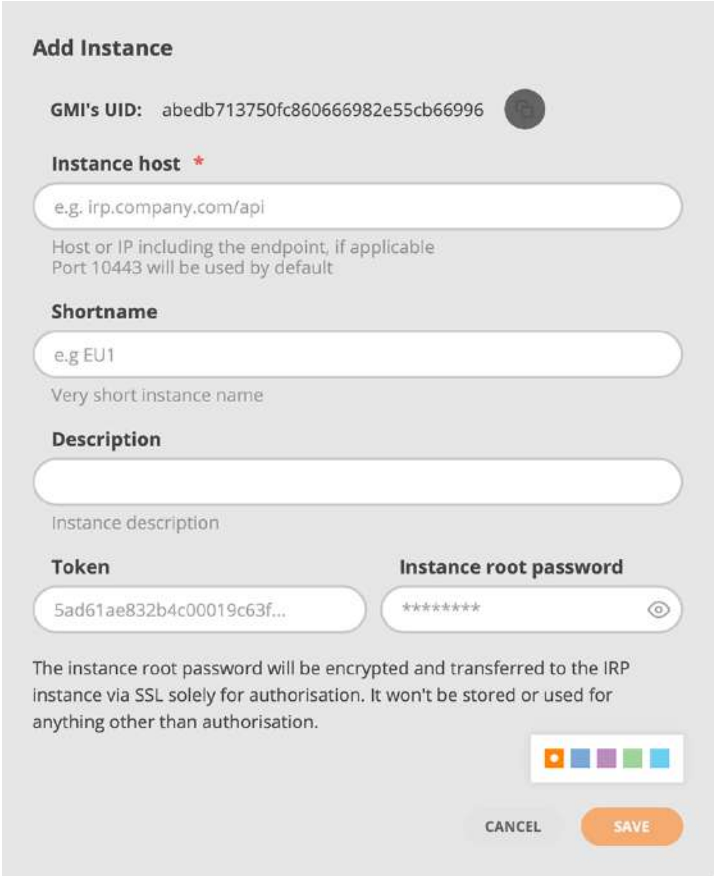

Figure 3.13.2: Adding an instance

3.13.2 User Management

The User Management tab is accessible under the Management main menu section.

Users who can access GMI are managed in User Directories. User directory is a generic name for an external user management provider for example LDAP (either POSIX or Active Directory). GMI can interface with as many user directories as needed in a specific environment.

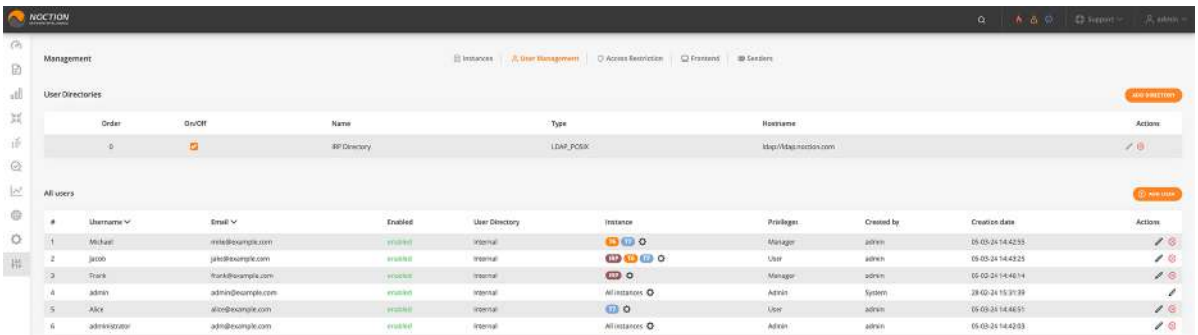

Figure 3.13.3: User Management

3.13.2.1 User token management

Along with GMI tokens, the user tokens can also be added for each particular user. This serves as an additional security enhancement for users who would like to integrate IRP into their own monitoring systems or reporting tools.

User token generation can be performed only by the logged user for himself, however instance administrators are able to remove tokens for users who are inactive or should not have any tokens added.

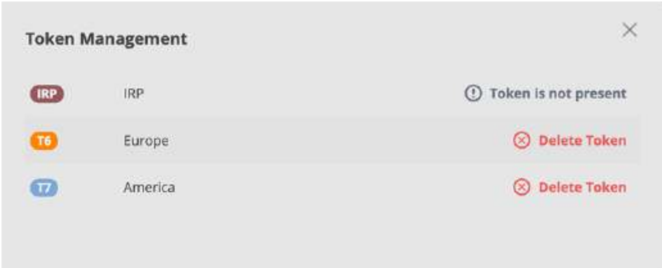

Figure 3.13.4: User token management

3.13.2.2 Internal user directory

GMI includes an Internal user directory that allows multiple user accounts to be setup.

To add a new user, one should click the ADD USER button and fill out the corresponding fields in a popup window.

There are 3 user roles available in GMI, with specific limitations and privileges:

- User - cannot manage other users or instances
- Manager - cannot manage users
- Admin - full privileges

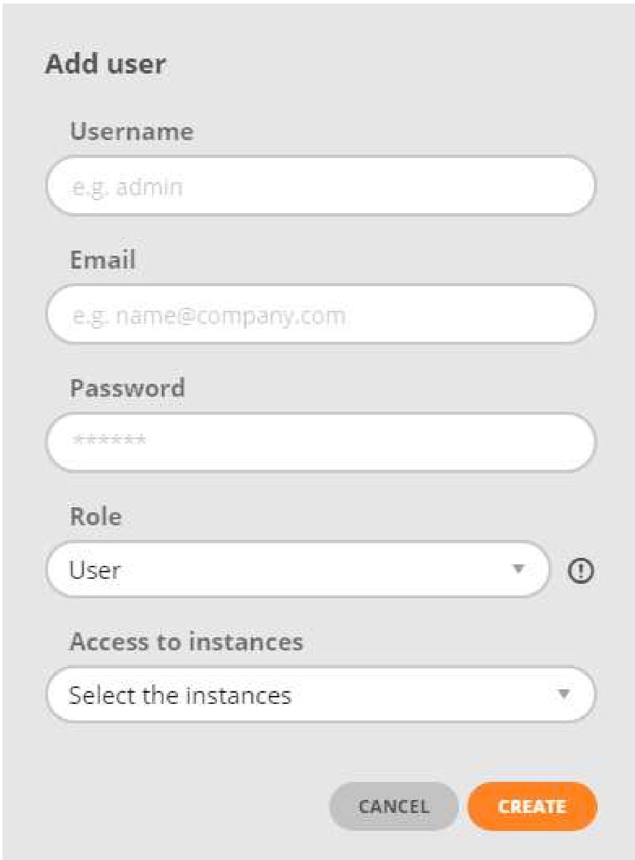

Figure 3.13.5: Adding users

3.13.2.3 LDAP and Active Directory

LDAP or AD user directories can be added, updated and removed from GMI by accessing "Management→User Management" tab. Each user directory takes a series of parameters specific for the protocol.

⚠ All operations with DNs (initial bind DN, group DNs, user names) are case insensitive and also strip redundant whitespace.

Refer individual protocol documentation for how to correctly configure one or another user directory.

The example below offers a generic set of parameters required to configure GMI to use Active Directory for access management.

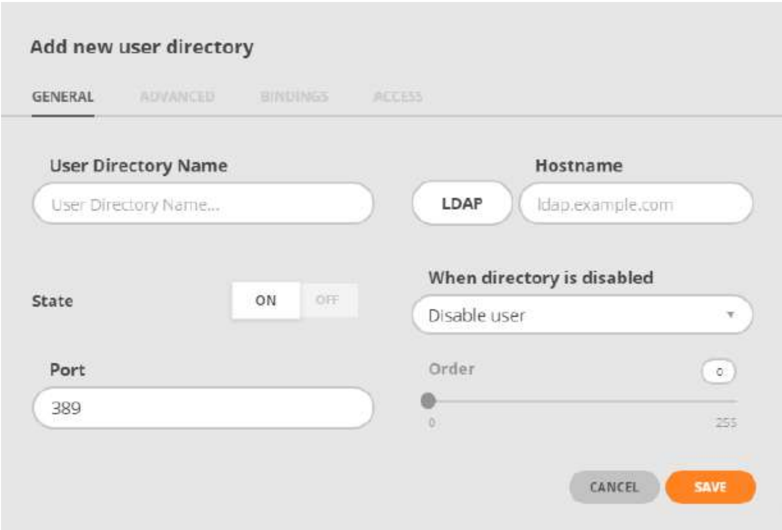

Figure 3.13.6: User Directory configuration

The general tab covers:

- User directory name - the name assigned to the directory within GMI,
- User directory hostname in the form of either IP address or domain name (LDAP/LDAPS),
- Enabling or disabling a user directory,
- The option to disable or remove users completely for a disabled directory.
- User directory port
- Order specifies when this user directory will be examined by GMI compared to other user directories,

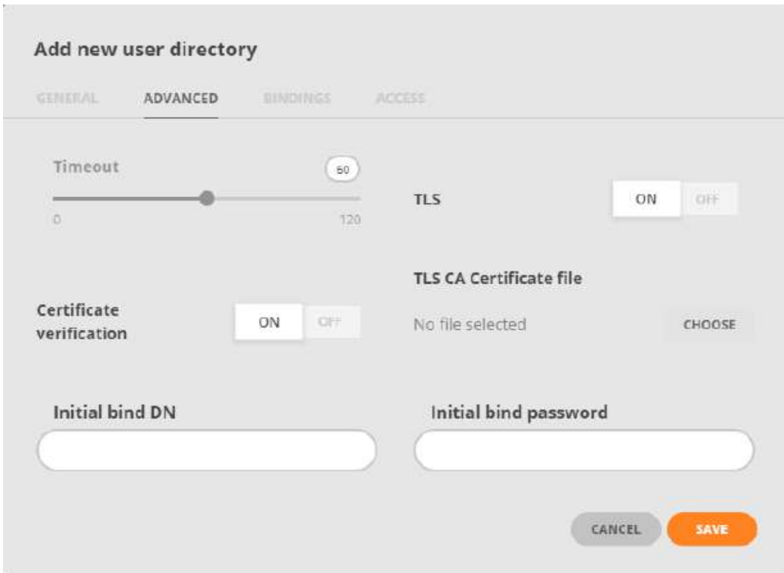

Figure 3.13.7: User Directory Advanced Options

Advanced configuration of a user directory covers:

- Timeout before failing a connection to this user directory
- TLS use
- Certificate verification
- CA certificate used to verify server's certificate in case the Certificate verifications is turned on
- Initial binding user name that GMI uses to authenticate itself
- Initial bind password assigned to GMI.

Usually bind passwords are not required to access directories. If a password is required and configured via GMI Frontend, it will be scrambled in GMI configuration files. If the password is specified directly in GMI configuration it is provided in clear-text with the condition that it does not start/end with scrambled password encapsulation characters.

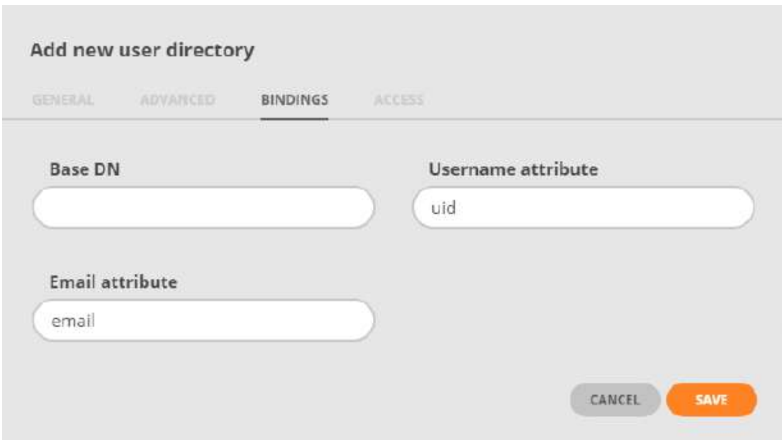

Figure 3.13.8: User Directory Bindings

Bindings maps User Directory attributes to GMI specific attributes, for example:

- Base DN specifies the root distinguished name and user subtree

GMI recognizes BOTH short and full user identifiers. Examples below are both valid directory entries that will match user "chris" with long name "Mr. Chris Smith":

```
cn: ops
uniqueMember: chris
```

and

```
cn: ops
uniqueMember: cn=Mr. Chris Smith,ou=employees,ou=People,dc=ops,dc=org
```

- Username and Email fields map User Directory attributes to GMI user attributes,

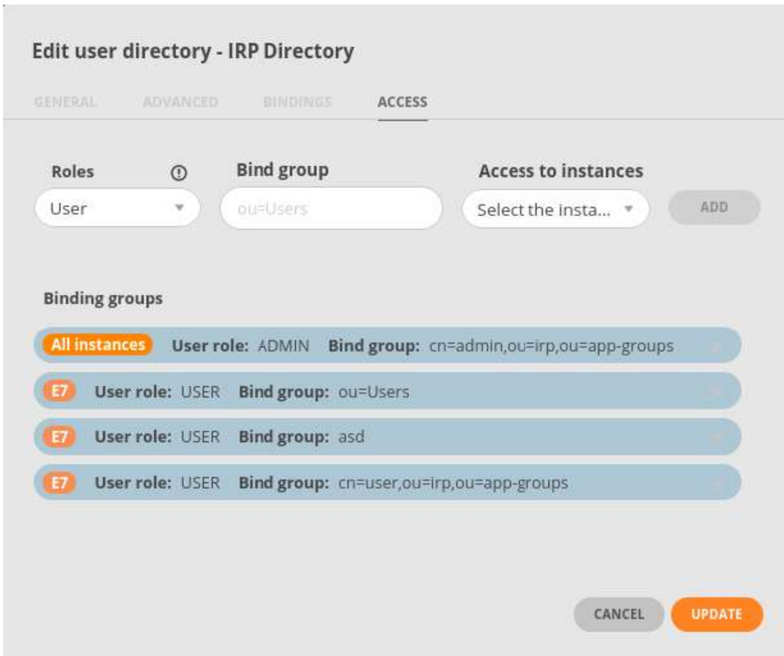

Figure 3.13.9: User Directory Access Options

Access configuration of a user directory covers:

- Roles assigned (either User, Manager or Admin),
- Bind group - the name of the attribute that uniquely identifies a given group or user.

Do NOT provide the Distinguished Name - the name that includes an object's entire path to the root of the LDAP namespace. Introduce the Relative Distinguished Name instead - an object name without a path, or a partial path (Example: "cn=support").

- Access to IRP instances available in GMI. (Users with the Admin roles have access to all IRP instances)

3.13.3 Access Restriction

This section covers restrictions on the ranges of IP addresses that can access GMI's Frontend and an option to add an SSL Certificate.

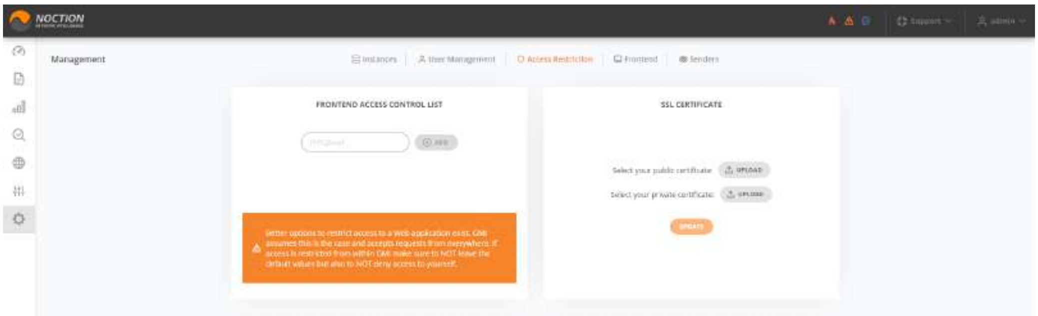

Figure 3.13.10: Access Restriction

3.13.4 Frontend

Custom Login screen and Navigation Bar logo images can be uploaded by using the Upload buttons. The new settings are applied after clicking the "SAVE" button.

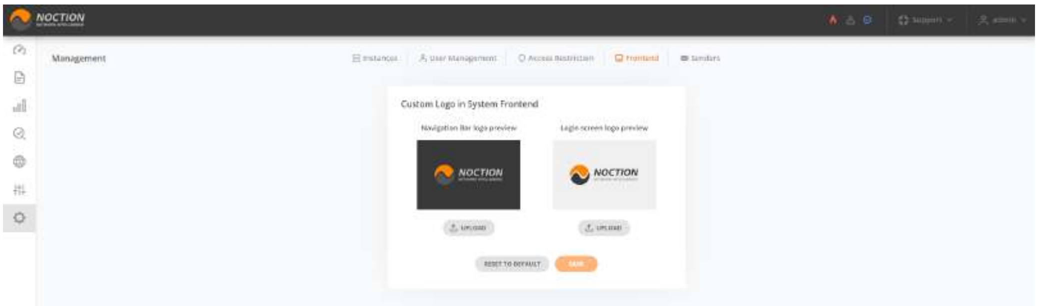

Figure 3.13.11: Frontend

3.13.5 Senders

To receive GMI reports and graphs via email or Slack, one should setup GMI senders. Click the "ADD A SENDER" button, fill out the required fields and click save.

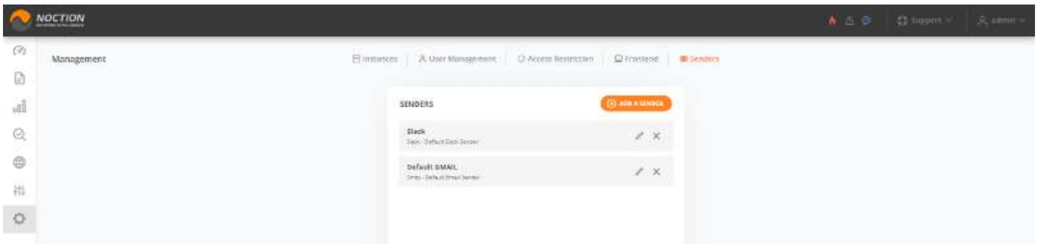

Figure 3.13.12: Senders

3.14 Emailing subscription

Proceed to Username→Emailing and sign up for the reports and graphs emails. Edit existing subscriptions or delete them in this view as well.

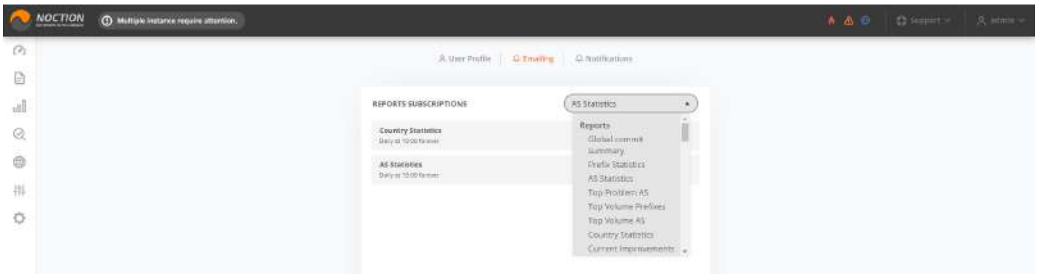

Figure 3.14.1: Reports and Graphs emailing subscription

Select the desired report or graph. Fill out the required fields and click Save.

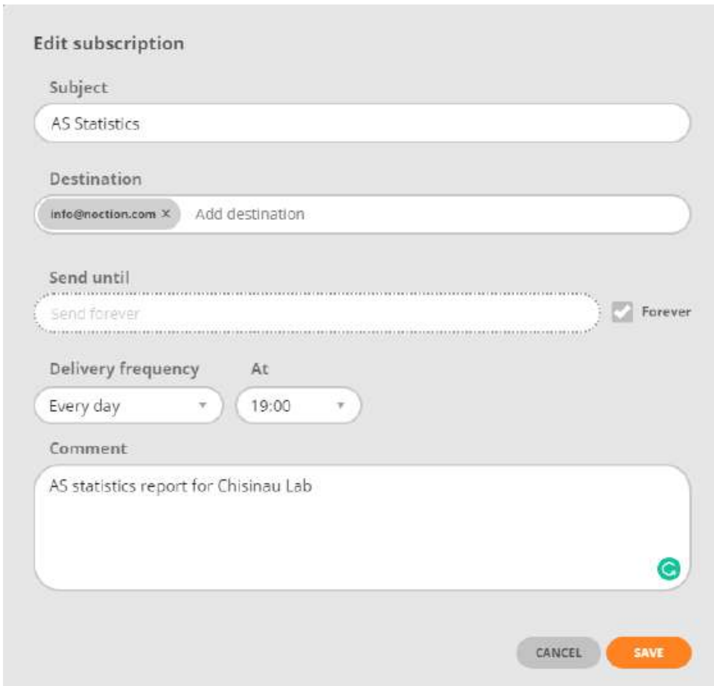

Figure 3.14.2: Emailing subscription details

3.15 Events

The system events are diagnostic messages that were sent by the Components of IRP instances, as well as notifications caused by the BGP monitoring algorithm. The events are displayed in the status bar located at the top of the system frontend (See Login into GMI). A specific IRP instance should be selected to display the corresponding results. Several or all the events can be marked as "Read", so they will no longer be displayed in the status bar.

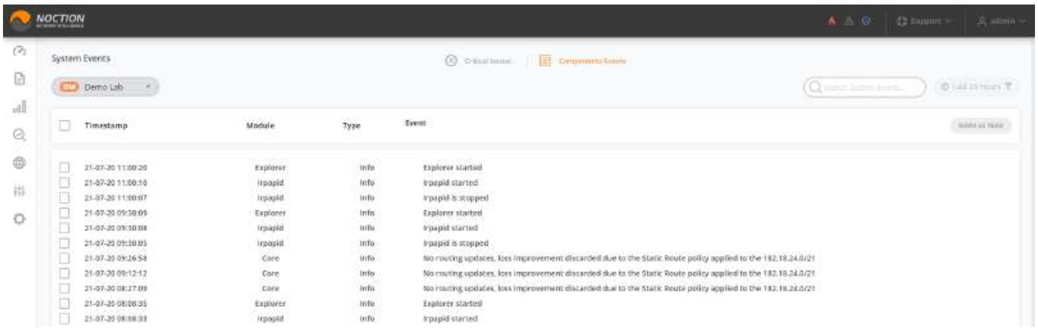

Figure 3.15.1: System events

3.16 Notifications and Events

IRP instances produce a huge number of various events and some of them are critical for administrator's awareness. Notifications in GMI allow administrators to subscribe to any of the available IRP instances generated events using the following channels:

- SMS
- Email
- Slack (via Webhook)
- SNMP Traps

IRP service Irppushd provides this feature. In order for Notifications to be delivered correctly the corresponding channel configuration shall be provided. By default only email notifications can be delivered since IRP uses the embedded system email service to send them.

More so, users should subscribe for specific events.

Only events for valid subscriptions using correctly configured channels will be delivered.

Refer section IRP instance Notifications for details about configuring, subscribing and contents of notifications.

Refer section Notification and events for details about individual configuration parameter.

3.16.1 Events

The list of events monitored by IRP that can generate notifications is provided below.

When one of the IRP components detects a transition form normal to abnormal traffic behavior or back it fires these events:

- Abnormal correction: irpflowd
- Abnormal correction: irpspand
- Inbound traffic low: SPAN
- Inbound traffic low: Flow
- Inbound traffic normal: Flow
- Inbound traffic normal: SPAN
- Outbound traffic low: SPAN
- Outbound traffic low: Flow
- Outbound traffic normal: Flow
- Outbound traffic normal: SPAN

When Commit Control limits are exceeded per provider or overall one of the following events fires. Refer section 4.12 for configuring the actual limits of the events.

- Commit Control overload by X Mbps
- Commit Control overload by X%
- Commit Control provider X overloaded by Y Mbps
- Commit Control provider X overloaded by Y%

When an IRP component (re)loads the configuration it validates it and depending on results fires one of the following events:

- Configuration Invalid: Bgpd
- Configuration Invalid: Core
- Configuration Invalid: Explorer
- Configuration Invalid: Irpapid
- Configuration Invalid: Irpflowd
- Configuration Invalid: Irpspand
- Configuration Ok: Bgpd
- Configuration Ok: Core
- Configuration Ok: Explorer
- Configuration Ok: Irpapid
- Configuration Ok: Irpflowd
- Configuration Ok: Irpspand

Outage detection algorithm fires one of the following events when it confirms congestion or outage problems and reroutes traffic around it:

- Congestion or Outage
- Outage: Confirmed and rerouted

Explorer periodically checks the PBRs and its expected probing performance and triggers the following events:

- Failed PBR (IPv6) check for provider
- Failed PBR (IPv4) check for provider
- Successful PBR (IPv4) check for provider
- Successful PBR (IPv6) check for provider
- Explorer performance low
- High number of VIP prefixes degrades IRP performance

IRP BGP Internal and External monitors fire the following events:

- ExternalMonitor (IPv4) Failed status for a provider. All improvements towards the provider will be withdrawn.
- ExternalMonitor (IPv4) OK status for a provider. All improvements towards the provider will be announced.
- ExternalMonitor (IPv6) Failed status for a provider. All improvements towards the provider will be withdrawn.
- ExternalMonitor (IPv6) OK status for a provider. All improvements towards the provider will be announced.
- InternalMonitor (IPv4) Failed status for a provider. All improvements towards the provider will be withdrawn.

- InternalMonitor (IPv4) OK status for a provider. All improvements towards the provider will be announced.
- InternalMonitor (IPv6) Failed status for a provider. All improvements towards the provider will be withdrawn.
- InternalMonitor (IPv6) OK status for a provider. All improvements towards the provider will be announced.

When statistics collection over SNMP is up or down IRP fires the following events:

- Provider SNMP stats down: X
- Provider SNMP stats up: X

Bgpd raises these events when BGP sessions are established/disconnected:

- IRP BGP session disconnected
- IRP BGP session established

When IRP identifies conditions to re-route traffic (make an improvement) and additionally it considers the differences to be excessive it raises these events:

- Excessive packet latency for prefix
- Excessive packet loss for prefix
- Improvements spike
- Low rate of announced IPv4 improvements
- Low rate of announced IPv6 improvements
- New improvement

Once an IRP component is started, stopped or restarted it raises the following events:

- Service started: Bgpd
- Service started: Core
- Service started: Explorer
- Service started: Irpapid
- Service started: Irpflowd
- Service started: Irpspand
- Service stopped: Bgpd
- Service stopped: Core
- Service stopped: Explorer
- Service stopped: Irpapid
- Service stopped: Irpflowd
- Service stopped: Irpspand

3.16.2 SNMP Traps

SNMP traps is a widely used mechanism to alert about and monitor a system's activity.

IRP SNMP traps not only notify about some IRP platform event but also include the list of varbinds which contain detailed information related to the thrown trap. The complete list of traps and varbinds with their descriptions can be found at `/usr/share/doc/irp/NOCTION-IRP.mib`

3.16.3 Notifications

Notifications are sent only if a valid subscription has been created. Subscriptions, similar to regular emailing of reports can be found under Username→Notifications menu.

Notifications page provides an overview of existing subscriptions with features to create, edit, delete them. Check the following figure for a preview.

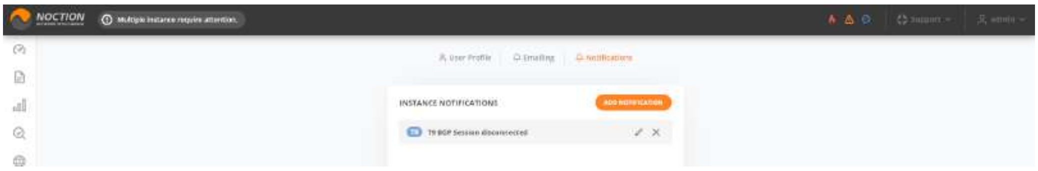

Figure 3.16.1: Notifications

Use Add new subscription or Events drop-down to create a new subscription or edit, delete existing subscriptions by clicking corresponding icons. Creating or editing a subscription will bring up a pop-up like the one in the following figure.

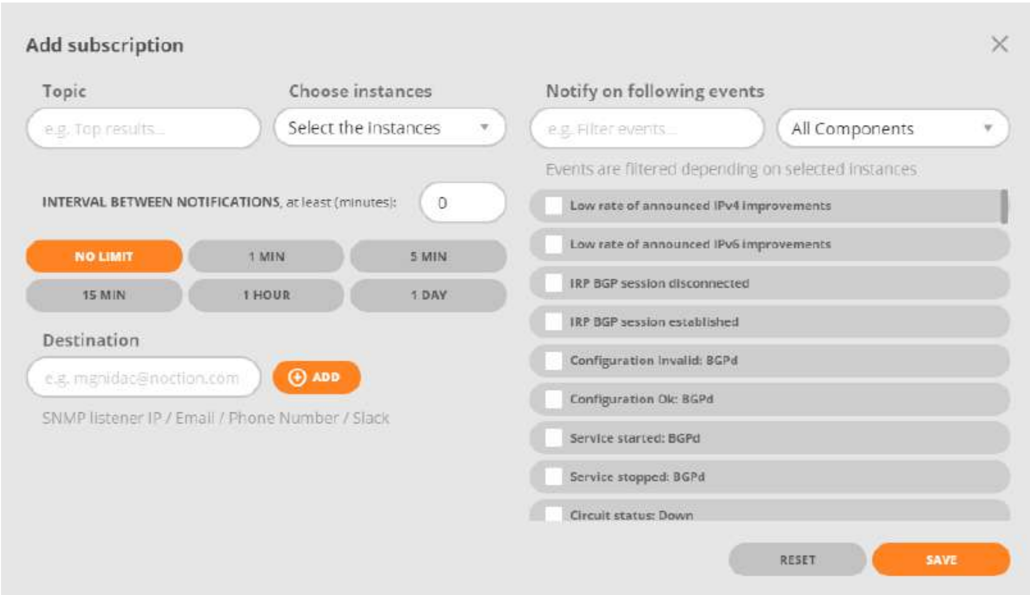

Figure 3.16.2: Subscribe notifications

Subscription details include:

- Topic - the title given to subscription in Notification page and also a textual part of notifications

> If SMS notifications are used it is advised to keep the topic short so that there is enough space left to include details about the event

- Choice of one or more IRP instances
- Interval between notifications - sets a rate limit of how often an email and/or SMS notification is sent. Value 'No limit' for the interval depicts no rate limits and all events will trigger a notification

> SNMP Traps notification are not constrained by the interval between notifications. SNMP Traps are sent immediately as they are raised

⚠ The rate limit is enforced per subscription so it is still possible to receive multiple notifications if the subscribed events are part of different subscriptions

- Destinations - IP address of a trap receiver for SNMP Trap notifications, email addresses, phone numbers and webhook channels. Multiple destinations of same type can be provided. At least one valid destination must be provided per subscription

ℹ IRP parses and recognizes whether a provided destination is a valid IP address, email address, phone number or web hook channel.
Supported formats are:

email name@host.domain

phone +digits

webhook #channel_name

snmptrap ipv4 or ipv4:port or [ipv6] or [ipv6]:port

- Event filters - allows filtering of events by textual description or by IRP component. Remove the filters to see all the subscribed events

- Subscribed events - the full list of events supported by IRP and tick marks for the events in this subscription. At least one event is required for a valid subscription.

3.17 User Profile

One can access Profile details by navigating to <Username> → Profile in the top right menu. This section allows to see the general profile details and preferences including the user's email, date and time formats and browser notifications permission. There is also an option to change password and see the most recent active sessions details.

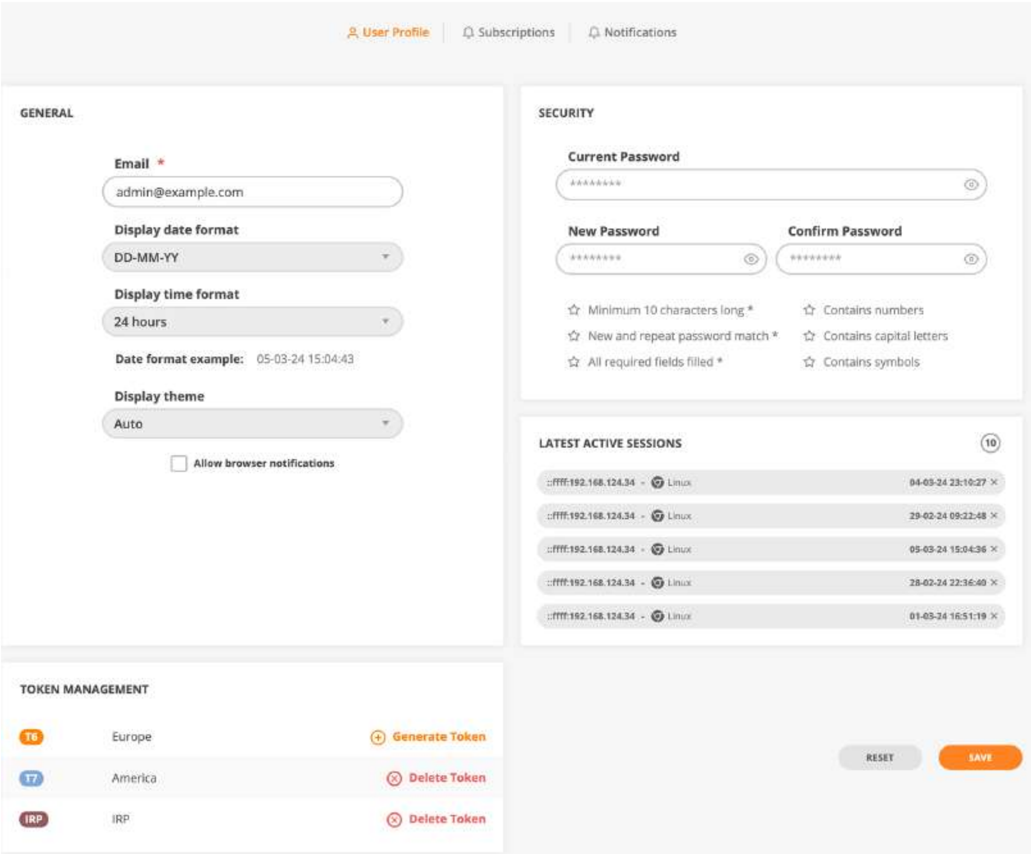

Figure 3.17.1: User Profile

3.18 Maintenance windows

Maintenance works are on everybody's agenda in current fast paced and continuously evolving networks. During maintenance network engineers are very busy and will welcome any help their systems can offer in carrying out those works with the least amount of headaches. IRP instances are clearly not in the top of network engineer's priorities and asking to suspend or shutdown providers immediately before a maintenance window starts and restart the provider back once the maintenance works end is not very helpful if not even annoying.

GMI offers the facility to plan maintenance windows in advance for providers on the connected IRP instances. Knowing when a maintenance window starts and ends, the IRP instances exclude specific provider links from either performance optimization or bandwidth control. More so, there is a capability to reshape the traffic flowing in and out of a network to anticipate any downtime on a link.

Properly configured maintenance windows allows IRP instnces the time to move most of the outbound traffic and deflect most of inbound traffic away from the provider link that is scheduled for maintenance. Having only a small fraction of traffic or none at all on the maintenance link before the downtime starts avoids any (shall we say, catastrophic) spikes, possible overloads and consequently unpredictable behavior of the remaining live network equipment.

Specifically the following applies:

- a maintenance window is configured in advance and can be removed/revised at any time
- a maintenance window sets details for single provider. If needed multiple maintenance windows can be setup and even overlapping maintenance windows are OK
- GMI highlights maintenance windows in the Frontend, so that it is easy to spot current maintenance window status

- optionally GMI can instruct IRP instances to preserve existing improvements so that once the maintenance window ends improvements are reimplemented. It is advised that this feature is used only when the maintenance window is very short (a few minutes long)
- an unloading period can be setup. During unloading IRP instances actively re-route outbound prefixes through other available providers. While IRP instances are able to make most of the unloading improvements fast, consideration shall be given to the announcement rate limitations setup in Bgpd in order for all the improvements to reach network routers in time for maintenance window starting time
- a prepend time can be setup. This is only applicable if Inbound optimization is operational. If this time is setup then an IRP instance will prepend configured inbound prefixes with the maximum allowed number of prepends through the provider link under maintenance in order to deflect inbound traffic towards other providers.

To review Maintenance windows, navigate to the corresponding left side bar menu option.

Figure 3.18.1: Maintenance windows menu button

The list displays all the current and future maintenance windows configured for providers in various IRP instances.

Figure 3.18.2: Maintenance windows

As shown in the screen-shot above, the list highlights:

- Current color-coded status of the maintenance window. Red and Orange depict unloading and actual maintenance phases while Blue and Green depict future planned windows and past finished windows correspondingly
- Begin and End date-times of maintenance windows
- Provider under maintenance
- Withdraw Improvement option enables or disables withdrawal of existing Outbound Improvements to that provider
- Seconds to Unload defines time interval in seconds prior to beginning of the maintenance window when a particular IRP instance starts to unload outbound traffic from provider

⚠ If Seconds to Unload/Seconds to Prepend time intervals set to zero the IRP instance does not perform corresponding action

When seconds to unload is specified it should be at least 5 minutes (300 seconds) long with larger time intervals allowing IRP more time to assess and reroute traffic flowing through provider with scheduled maintenance window. Very short time intervals might not have the desired effect since usually an IRP optimization cycle takes 2-3 minutes to finish

- Seconds to Prepend defines time interval in seconds prior to beginning of the maintenance window when IRP announces all inbound prefixes with maximum prepends to provider in order to deflect inbound traffic towards other providers.

Maintenance windows can be added, edited or deleted using the designated action buttons.

A maintenance window is added by specifying the following:

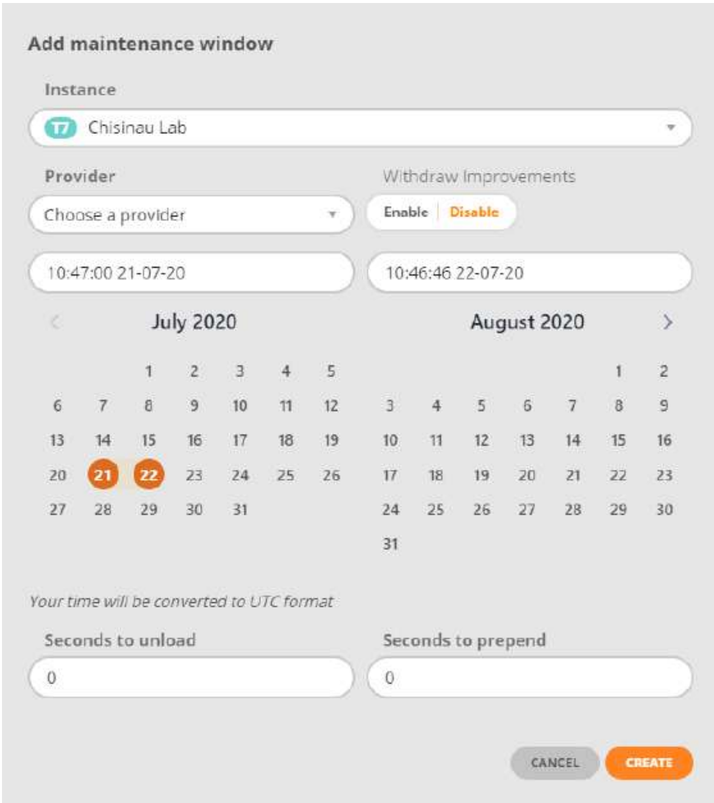

Figure 3.18.3: Maintenance window popup

Chapter 4

Configuration parameters reference

4.1 Database credentials

4.1.1 db.clickhouse.dbname

Name of the ClickHouse database that contains the IRP tables.

Default value: `irp`

4.1.2 db.clickhouse.host

Database host. ClickHouse server host name or IPv4/IPv6 address.

Default value: `127.0.0.1`

4.1.3 db.clickhouse.http_port

TCP port number used to connect to the ClickHouse server via REST API.

Possible values: `1-65535`

Default value: `8123`

4.1.4 db.clickhouse.password

The password used to connect to a ClickHouse server.
The password will be changed to a random one by the IRP installation process.

Default value: `irp`

4.1.5 db.clickhouse.port

TCP port number used to connect to the ClickHouse server via native API.

Possible values: `1-65535`

Default value: `9000`

4.1.6 db.clickhouse.username

User name used to connect to a ClickHouse server.

Default value: `irp`

4.1.7 db.dbname

Name of a MySQL database that holds the IRP tables.

Default value: `irp`

4.1.8 db.host

Database host. MySQL server host name or IPv4/IPv6 address.

Default value: `localhost`

4.1.9 db.ourhost

IRP host. IRP box host name or IPv4/IPv6 address.

Default value: `localhost`

4.1.10 db.password

The password used to connect to a MySQL server.
The password will be changed to a random one by the IRP installation process.

Default value: `irp`

4.1.11 db.port

TCP port number to use for the connection to a MySQL database.

Possible values: `1-65535`

Default value: `3306`

4.1.12 db.username

User name used to connect to a MySQL server.

Default value: `irp`

4.2 Global parameters

4.2.1 global.agg_ipv4_max

Defines the maximum IPv4 aggregate mask for the improved prefixes advertised by Bgpd. If global.aggregate is enabled, IRP will not announce any prefix with the mask that's less than the mask defined in global.agg_ipv4_max.

Possible values: `8-24`

Default value: `16`

4.2.2 global.agg_ipv6_max

Defines the maximum IPv6 aggregate mask for the improved prefixes advertised by Bgpd. If global.aggregate is enabled, IRP will not announce any prefix with the mask that's less than the mask defined in global.agg_ipv6_max.

Possible values: `24-56`

Default value: `32`

4.2.3 global.aggregate

If this parameter is enabled IRP analyzes entire aggregates. When one can be improved the aggregate will be announced by Bgpd (refer to bgpd.updates.split, bgpd.peer.X.updates.limit.max). Largest possible prefixes are used when aggregates exceed global.agg_ipv4_max & global.agg_ipv6_max parameters for IPv4 and IPv6 accordingly. An aggregate dictionary is maintained by IRP in order to support this capability. The aggregate dictionary is periodically populated by `refresh_asn` or Bgpd (if Bgpd has sufficient information, by means of at least one full-view BGP peering session).

Disabling this parameter instructs IRP to operate with the smallest possible disaggregated prefixes (/24 for IPv4 and /48 IPv6).

Switching this parameter ON/OFF might leave a large number of small prefixes in the aggregate dictionary with undesirable side effects. The contents of the dictionary should be reviewed and refreshed in case it lists undesirable prefixes. To prevent issues IRP improvements should be cleared before applying this change. Clearing improvements ensures that disaggregated prefixes announced by IRP are not present on the network.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.2.4 global.bw_overusage

Enables or disables automatic Flowspec throttling policies for prefixes with excessive bandwidth spikes.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.2.5 global.exchanges

Defines the path to the Exchanges configuration file.

Default value: `/etc/noction/exchanges.conf`

Recommended value: `/etc/noction/exchanges.conf`

4.2.6 global.exchanges.auto_config_interval

Defines the Internet Exchanges auto re-configuration interval in seconds. Auto re-configuration is applied to providers of type Internet Exchange with enabled auto re-configuration. Refer peer.X.auto_config.

Possible values: `3600-2678400`

Default value: `86400`

4.2.7 global.failover

Enables and disables failover feature.

> Enabling failover requires extensive preparatory activities. Refer IRP Failover, Failover Configuration, Setup Failover wizard for details.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.2.8 global.failover.log

Path to failover log file.

Default value: `/var/log/irp/failover.log`

4.2.9 global.failover.use_communities

This parameter allows monitoring presence of improvements from master instance by using communities specified in bgpd.peer.X.master_communities (refer bgpd.peer.X.slave_communities).

Master and slave communities in a BGP peering should be unique if the parameter is enabled.

If a network setup requires the same community to be used on both master and slave instances then the parameter should be disabled and failover shall use bgpd.peer.X.master_localpref only.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.2.10 global.failover_identity_file

Defines the path to failover master identity file. Set this parameter only when a SSH key file other than ~/.ssh/id_rsa is used. Refer global.failover.

Possible values: `Path to identity file`

4.2.11 global.failover_role

Defines the role of the IRP instance in a failover configuration. Known roles are Master and Slave. Used only when failover is enabled. Refer global.failover.

Possible values: `0 (Master), 1 (Slave)`

Default value: `0`

4.2.12 global.failover_slave.ip

Defines the IPv4 address of the slave IRP instance used for configuration sync in a failover configuration. Used only when failover is enabled. Refer global.failover.

Possible values: `Valid IPv4 address`

Example value: `10.10.0.2`

4.2.13 global.failover_slave.ipv6

Defines the IPv6 address of the slave IRP instance used for configuration sync in a failover configuration. Used only when failover is enabled. Refer global.failover.

Possible values: `Valid IPv6 address`

Example value: `2001:db8::ff00:42:8329`

4.2.14 global.failover_slave.port

Defines the SSH port of the slave node in a failover configuration. Used only when failover is enabled.
See also global.failover.

Possible values: `1-65535`

Default value: `22`

4.2.15 global.failover_timer_fail

Defines the period of time in seconds during which master is considered alive before the slave becomes active in a failover configuration. Used only when failover is enabled.
See also global.failover.

Possible values: `30-3600`

Default value: `300`

4.2.16 global.failover_timer_failback

Defines the period of time in seconds for slave to switch back to standby after it detects master became alive in a failover configuration. Used only when failover is enabled.
See also global.failover.

Possible values: `30-3600`

Default value: `300`

4.2.17 global.flowspec

Enables/disables Flowspec capability globally.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.2.18 global.flowspec.pbr

Enables/disables use of Flowspec policies instead of Policy Based Routing. This is available only when Flowspec is enabled. Refer global.flowspec.

Possible values: `0 (Disabled), 1 (Enabled in intrusive), 2 (Enabled always)`

Default value: `0`

4.2.19 global.frontend_acl

Defines whether access to the IRP Frontend must be restricted and controlled by an ACL. See also global.frontend_acl_ips.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.2.20 global.frontend_acl_ips

Defines the IPs or networks that should be allowed to access the IRP Frontend.

Possible values: `Valid IPv4, IPv6 address or subnet definition in CIDR format`

Default value: `0.0.0.0/0 ::/0`

4.2.21 global.ignored.asn

Defines the list of ASNs to be ignored by IRP.
Format:

1. Space-separated list of AS numbers that must be ignored by IRP.
2. Absolute path to a newline-separated file containing a list of AS numbers that must be ignored by IRP.

This parameter should list all AS numbers that must be ignored by Irpspand, Irpflowd, Explorer and the Core. No improvements will be performed by the Core for prefixes that are announced by ASNs listed within this parameter. No data will be gathered by Irpspand/Irpflowd for any source or destination IPv4/IPv6 address that are announced by ASNs that are listed within this parameter. No probes will be sent by Explorer to any destination IPv4/IPv6 address that are announced by ASNs listed within this parameter.
Refer also global.ignored_communities, global.ignorednets.

Possible values: 1 - 4294967295.

4.2.22 global.ignored.unannounced

Prefixes which aren't present in a BGP routing table wouldn't be analyzed nor optimized by Outbound optimization algorithms.

As example, spoofed IP addresses may not belong to advertized prefixes but the traffic itself may have significant volume and be optimized by IRP.

Therefore, in such a case the recomendation is to enable the parameter to do not optimize traffic from unkjnown origins.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.2.23 global.ignored_communities

Defines the list of BGP Community attributes the network uses to mark prefixes that IRP must ignore. IRP monitors routes marked with at least one of these BGP Community attributes and dynamically updates a list of prefixes to ignore. The decision what prefixes to mark with one of these attributes is applied on the network and network operators do not need to be explicitly set in IRP too.

This (dynamic) list of prefixes will be ignored by Irpspand, Irpflowd, Explorer and the Core. No improvements will be performed by the Core for such prefixes. No data will be gathered by Irpspand/Irpflowd for any source or destination IPv4/IPv6 address within such prefixes. No probes will be sent by Explorer to any destination IPv4/IPv6 address within such prefixes.

Refer also global.ignored.asn, global.ignorednets.

Possible values: `valid BGP community attribute list`.

4.2.24 global.ignorednets

Defines the list of networks to be ignored by IRP.

Format:

1. Space-separated list of local IPv4/IPv6 prefixes that should be ignored by IRP.
2. Absolute path to a newline-separated file containing a list of IPv4/IPv6 prefixes that should be ignored by IRP.

If netmask is not clearly specified, the system assumes /32 for IPv4 addresses, and /128 for IPv6 addresses.

> 224.0.0.0/3 is always ignored.
> - IPv6 probing is performed only to 2000::/3 address range (see: IPV6 Address Space)
> - All IPv4/IPv6 addresses assigned to IRP server are automatically added to ignored networks

This parameter lists all IPv4/IPv6 addresses that should be ignored by Irpspand, Irpflowd, Explorer and the Core. No improvements will be performed by the Core for /24 subnets listed within this parameter as /24 or a less specific network. No data will be gathered by Irpspand/Irpflowd for any source or destination IPv4/IPv6 address listed within this parameter. No probes will be sent by the Explorer to any destination IPv4/IPv6 address listed within this parameter.

Refer also global.ignored.asn, global.ignored_communities.

Possible values: See above.

Default value: `127.0.0.0/8 10.0.0.0/8 169.254.0.0/16 172.16.0.0/12 192.168.0.0/16 100.64.0.0/10 203.0.113.0/24 198.18.0.0/15 192.0.0.0/24 192.0.2.0/24 2001:db8::/32`

Recommended value: See default value.

> Recommended networks to ignore are:
> - networks from: Section 3 of RFC5735, as listed in the default value above
> - networks from: Section 2 of RFC3849

4.2.25 global.improve_mode

Defines the IRP operating mode (see IRP Optimization modes).

This parameter adjusts the priorities and rules for prefix improvements. Prefixes can be improved in three different ways:

1. Performance optimization: Decrease loss, then decrease latency;
2. Cost optimization: Decrease loss, then decrease cost while keeping the latency within the preconfigured level;

Loss, cost and latency are improved in strict order, depending on the selected operating mode.

Possible values: `1 (Performance), 2 (Cost)` (see above).

Default value: `1`

See also: Commit Control, core.commit_control, peer.X.cc_disable

4.2.26 global.inbound.injection

Defines how IRP is used to inject Inbound improvements.
In "Pull" mode an external script is used to pull Inbound improvements from IRP API and re-configure routers' access lists.
Example script `/usr/bin/irpTransitPull.pl` could be used to work in the Pull mode.

Possible values: `0 (PULL), 1 (BGP)`

Default value: `1`

4.2.27 global.inbound_conf

Defines path to file with configured inbound prefixes.

Possible values: `path to file`

Default value: `/etc/noction/inbound.conf`

Recommended value: `/etc/noction/inbound.conf`

4.2.28 global.inbound_transit

Enables or disables inbound optimization of transiting traffic. Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.2.29 global.ipv6_enabled

Defines whether IPv6 is enabled in the system. Currently used for the Frontend only. Even if IPv6 is enabled via this parameter, other components configuration must be adjusted for IPv6 as well.

IRP prior to 3.3-2 allows configuration of both IPv4 and IPv6 sessions on a single router, while IRP 3.3-2 requires definition of two separate BGP sessions. It is recommended to split BGP sessions in the form of two different routers before upgrading to 3.3-2, otherwise IRP configuration will not be valid. Make sure the newly added router is linked to corresponding providers to ensure IPv6 optimization works properly. InternalMon for IPv6 session monitoring requires correct configuration of BGP MIB (IPv6) mode (see peer.X.mon.ipv6.internal.mode parameter). Currently support for Brocade, Cisco and Juniper MIBs is available.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.2.30 global.master_management_interface

Defines the management network interface. In most cases it is the same as the probing interface. When failover is enabled (global.failover) slave's management interface (global.slave_management_interface) must be configured too.

Possible values: any valid system network interface name

Default value: `eth0`

4.2.31 global.master_probing_interface

Defines the probing network interface. In most cases it is the same as the management interface. When failover is enabled (global.failover) slave's probing interface (global.slave_probing_interface) must be configured too.

> This parameter is used only for displaying operating system interface(s) status in Frontend. It does not actually configure Explorer behavior, which depends on Provider.

Possible values: any valid system network interface name

Default value: `eth0`

4.2.32 global.master_rd

Specifies the Routing Domain that hosts the master node of IRP in a failover configuration. By default RD=1 hosts IRP nodes.

> It is recommended that master node of IRP is hosted in RD=1 at all times.

Possible values: `1-100`

Default value: `1`

Recommended value: `1`

4.2.33 global.nonintrusive_bgp

Instructs the system to run in a non-intrusive BGP mode (see IRP Operating modes).
All improvements made in a non-intrusive mode, will not be automatically injected into the routers.

Possible values: `0 (Intrusive), 1 (Nonintrusive)`

Default value: `1`

Recommended value: `1` at first start, `0` after manual tests are performed and the system is ready to go into intrusive mode

4.2.34 global.offpeak_hour

Defines customer's network usual off-peak hour of the day.

Possible values: `0-23`

Default value: `3`

4.2.35 global.outbound

Enables or disables outbound optimisation.

⚠ Outbound optimization is disabled only for standalone Inbound optimization is used.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.2.36 global.outbound.performance

Enables or disables Outbound Performance optimisations.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.2.37 global.png.datadir

Defines the file-system directory path for storing image files (Graphs).

Default value: `/usr/share/irp/web/RRD`

Recommended value: `/usr/share/irp/web/RRD`

4.2.38 global.policies

Defines the path to the Routing Policies (1.2.9) configuration file.

Default value: `/etc/noction/policies.conf`

Recommended value: `/etc/noction/policies.conf`

4.2.39 global.policy.unpack_max

Limits the amount of unpacked prefixes per policy.
The parameter protects against specifying filters that are too broad.

Possible values: `1-100000`

Default value: `5000`

4.2.40 global.private_key

Public/Private key pair used to interoperate between IRP instances for purposes such as Global Commit Control.

Default value: `generated once during installation/upgrade`

4.2.41 global.public_key

Public/Private key pair used to interoperate between IRP instances for purposes such as Global Commit Control.

Default value: `generated once during installation/upgrade`

4.2.42 global.rd_rtt

Defines the latency distances between routing domains in the format rda:rdb:rtt where rda is the id assigned to one routing domain, rdb is the id assigned to the second routing domain and rtt represents the round trip time in miliseconds between them. rda and rdb must be different and assigned to providers. IRP assumes that distance from rda to rdb is equal to distance from rdb to rda.

The parameter takes a collection of such triplets that will define all the available inter-datacenter links between routing domains.

> It is important that the RTT value between routing domains is accurate. In case the value differs significantly from the correct value IRP will make improvement decision based on incorrect information and it will make unnecessary global improvements that will reroute more traffic via inter-datacenter links.

Refer peer.X.rd, peer.X.flow_agents, bgpd.rd_local_mark

4.2.43 global.rrd.age_max

Defines the maximum trusted interface load data age (seconds)

Data older than this interval will not be trusted by IRP. If interface rate for each provider link has not been updated for a specified amount of time, then IRP behavior will be changed as follows:

- If provider's limit_load is set, no further Cost/Performance improvements will be performed to that provider
- Commit Control will not perform further in/out improvements for this provider

Possible values: `120-240`

Default value: `120`

Recommended value: `120`. Should be increased only if frequent SNMP timeouts occur.

4.2.44 global.rrd.datadir

Defines the file system directory path for storing RRD database files.

Possible values: valid directory

Default value: `/var/spool/irp`

Recommended value: `/var/spool/irp`

4.2.45 global.slave_management_interface

Defines the management network interface for slave node in a failover configuration. In most cases it is the same as the probing interface. When failover is disabled (global.failover) the parameter is not used.

See also global.master_management_interface.

Possible values: any valid system network interface name

Default value: `eth0`

4.2.46 global.slave_probing_interface

Defines the probing network interface for the slave node in a failover configuration. In most cases it is the same as the management interface. When failover is disabled (global.failover) the parameter is not used.

See also global.master_probing_interface.

Possible values: any valid system network interface name

Default value: `eth0`

4.2.47 global.slave_rd

Specifies the Routing Domain that hosts the slave node of IRP in a failover configuration. By default RD=1 hosts IRP nodes.

Possible values: `1-100`

Default value: `1`

Recommended value: `1`

4.3 API daemon settings

4.3.1 apid.allowed_ips

This parameter defines the list of IPv4/IPv6 prefixes or addresses that are allowed to make requests to Irpapid.

⚠ By default, only server-local access to Irpapid is allowed. If there is a remote GMI installation, the IP address(es) of the GMI instance(s) should be added to this parameter.

Possible values: `list of IPv4/IPv6 prefixes`

Default value: `127.0.0.1 ::1`

4.3.2 apid.grpc.port

Specifies the TCP port on which the Irpapid is listening for interprocess communication.

Possible values: `1-65535`

Default value: `7602`

4.3.3 apid.listen.master_ip

Defines the IPv4/IPv6 address of the API. Allows IRP API calls to target another node on the network. If no value provided then localhost is used. When failover is enabled (global.failover) slave's listen IP (apid.listen.slave_ip) must be configured too.

Possible values: `list of IPv4/IPv6 addresses`

Default value: `::`

4.3.4 apid.listen.port

Defines the TCP port on which the irpapid is listening. If no value provided port 10443 is used.

Possible values: `1-65535`

Default value: `10443`

4.3.5 apid.listen.slave_ip

Defines the IPv4/IPv6 address of the API of the slave node in a failover configuration. Allows IRP API calls to target another node on the network. If no value provided then localhost is used. When failover is disabled (global.failover) slave's listen IP is not used.

See also apid.listen.master_ip.

Possible values: `list of IPv4/IPv6 addresses`

Default value: `::`

4.3.6 apid.log

Defines the file-system path to the iprapid log file.

Default value: `/var/log/irp/irpapid.log`

4.3.7 apid.log.level

Defines the logging level for the irpapid service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.3.8 apid.maxthreads

Defines the number of threads that irpapid is allowed to run. If no value provided 50 threads are used.

Possible values: `1-300`

Default value: `50`

4.3.9 apid.path.mtr

System path to MTR utility.

Possible values: `/valid/path`

Default value: `/usr/bin/mtr`

Default value varies depending on platform

4.3.10 apid.path.ping

System path to ping utility.

Possible values: `/valid/path`

Default value: `/usr/bin/ping`

⚠ Default value varies depending on platform

4.3.11 apid.path.traceroute

System path to traceroute utility.

Possible values: `/valid/path`

Default value: `/usr/bin/traceroute`

⚠ Default value varies depending on platform

4.4 Bgpd settings

4.4.1 bgpd.as_path

Defines the way Bgpd handles the AS-PATH attribute in the outgoing announcements. The selected options are evaluated in the configured order and the first valid AS-PATH will be used.

⛔ Note that IRP will refuse to announce improvements when all configured options fail to produce a valid AS-PATH. Option 0 allows announcements with empty AS-PATH but this is undesireable for other reasons noted below.

Keeping a correct AS-PATH can be a requirement for some NetFlow/sFlow processing since the provider AS or the destination AS for the corresponding flow record is taken from the BGP routing table.

Some installations use outgoing filters that allow empty AS-Path while redistributing iBGP routes to upstreams. In such cases, Bgpd must be configured not to advertise the improvements with an empty AS-PATH to prevent further redistribution of the improvements to upstream routers.

ℹ The reconstructed AS-path does not always correspond to the actual BGP AS-path.

"Use AS-Path from BMP" option is tied to the bgpd.improvements.remove.bmp_exact_aggregate parameter. Must be enabled and placed first in the list of AS path restore priority options for the improvements to be removed when an exact size prefix gets withdrawn from BMP.

Refer also to peer.X.ipv4.next_hop_as, peer.X.ipv6.next_hop_as, peer.X.aspath_for_ix, bgpd.as_path_borrowing, bgpd.improvements.remove.bmp_exact_aggregate.

Examples:

`bgpd.as_path=2 3` - this will instruct Bgpd to take first path from the database (reconstructed AS-Path). If that path is empty, then an AS-path with the provider's AS number and the prefix AS number should be composed.

`bgpd.as_path=3 0` - this will instruct Bgpd to compose an AS-path with the provider's AS number and the prefix AS number. If AS-Path remains empty, the improvements with an empty AS-Path are announced.

Possible values: Space separated list of options in the order of preference

- 0 - Allow empty AS-PATH
- 2 - Use non-empty reconstructed AS-PATH (Announce AS-path reconstructed from traceroute)
- 3 - Reconstruct AS path with provider ASN and prefix origin ASN
- 4 - Use AS-Path from BMP
- 5 - Use AS-Path from BGP Alternative paths (RFC 7911)

Second algorithm produces meaningful AS-PATH only when explorer.trace.all is enabled

Default value: `5 4 2 3`

4.4.2 bgpd.as_path_borrowing

Allows BMP to borrow AS path from other Provider with the same autonomous system number.

The parameter modifies only 4th algo ofbgpd.as_path parameter.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.4.3 bgpd.circuitissue.session_drop_delay_intervals

Defines the delay, in number of Bgpd scan intervals (bgpd.scaninterval), to wait before announcing a Circuit Issue Detection-initiated BGP session drop via FlowSpec.
The delay is required to allow prepend propagation before shutting down the provider's BGP session.

Possible values: `1-5`

Default value: `2`

4.4.4 bgpd.db.timeout.withdraw

Defines the time period (in seconds) before prefixes are withdrawn from the routing tables, after a database failure. This allows BGP daemon to function independently for a period of time after the IRP database becomes inaccessible due to a failure or manual intervention.

Possible values: `600-21600`(6 hours)

Default value: `14400`

Recommended values: `3600-14400`

4.4.5 bgpd.full_control

Sets default behavior of IRP in regards to inbound prefixes control and specifically:

- only announce improvements or
- only announce improvements and include all allowed providers or
- fully control inbound prefixes by always announcing to allowed providers.

This system wide default behavior can be overridden at inbound prefix level by specifying desired parameter value for inbound.rule.X.full_control.

Possible values: `0 (Improvements), 1 (If improved), 2 (All)`

Default value: `0`

4.4.6 bgpd.grpc.port

Specifies the TCP port on which the Bgpd is listening for interprocess communication.

Possible values: `1-65535`

Default value: `7605`

4.4.7 bgpd.improvements.remove.bmp_exact_aggregate

Removes an improvement when the exact prefix for such improvement is no longer seen in BMP and the AS-Path from BMP is in use.
See also bgpd.improvements.remove.withdrawn, bgpd.as_path.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.4.8 bgpd.improvements.remove.hold_time

Defines the time interval (in seconds) for deleting (from the IRP database) the improvements affected by "Remove on next-hop eq" or "Remove on aggregate withdraw" conditions
(see bgpd.improvements.remove.next_hop_eq and bgpd.improvements.remove.withdrawn).
This reduces the effects of route flapping to improvement cleanup.
Verification is performed on each BGP Scan, so that the values that are lower than the value of the bgpd.scaninterval parameter are not taken into account. The higher the values - the longer the time interval needed for improvements, which are still valid after aggregate route is withdrawn or on equal next-hop.
Bgpd does improvements check against the routers received and sent via iBGP in periodic time intervals. The process is referred to as the BGP Scan process.
Time interval between BGP Scannings can be configured in bgpd.scaninterval.

Possible values: `1-1000` seconds

Default value: `60`

Recommended values: `30-120`

4.4.9 bgpd.improvements.remove.next_hop_eq

Instructs Bgpd to remove a prefix from improvements when aggregate route's next hop has changed and points to the same next hop as the improvement.

This parameter shouldn't be enabled when bgpd.updates.split is disabled in a multi-routing domain configuration.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

Recommended value: `0`

4.4.10 bgpd.improvements.remove.withdrawn

Instructs Bgpd to remove prefix from improvements when aggregate is being withdrawn from the router. See also bgpd.improvements.remove.bmp_exact_aggregate.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

Recommended value: `1`

4.4.11 bgpd.improvements.strip_non_irp_communities

Instructs Bgpd to exclude from Updates of IRP improvements other communities except those configured in IRP.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.4.12 bgpd.log

Defines the complete path to the Bgpd log file

Possible values: `full path to log file`

Default value: `/var/log/irp/bgpd.log`

4.4.13 bgpd.log.level

Defines the logging level for the Bgpd service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.4.14 bgpd.mon.guardtime

Defines the Bgpd Monitoring guard time. All the controlled IP addresses must respond within the specified amount of time, for the provider to be restored from the FAIL state. In FAIL state, all improvements are withdrawn from that provider.
It is recommended that bgpd.mon.guardtime is set as a double value of the bgpd.mon.holdtime.

Possible values: `10-600`

Default value: `60`

Recommended value: `60`

4.4.15 bgpd.mon.holdtime

Defines the Bgpd Monitoring holdtime.

If any controlled IP address does not respond to all the requests during the holdtime, then corresponding provider enters the FAIL state. In FAIL state, all the improvements are withdrawn from that provider.

Possible values: `10-60`

Default value: `30`

Recommended value: `30`

4.4.16 bgpd.mon.internal.flap_guardtime

Defines time interval to protect from BGP session flapping.

Internal monitor will keep FAIL state until BGP session stays established within the configured period of time.

Possible values: `0-86400`

Default value: `0`

Recommended value: `600`

4.4.17 bgpd.mon.keepalive

Defines the Bgpd Monitoring keepalive interval (in seconds) between consequent ICMP Echo Requests to single controlled IP address.

Possible values: `1-10`

Default value: `10`

Recommended value: `5`

4.4.18 bgpd.mon.longholdtime

Defines the Bgpd Monitoring long holdtime.

If any controlled IP address does not respond to all the requests during the long holdtime, then corresponding provider enters the FAIL state. In FAIL state, all the improvements are withdrawn from that provider.

Possible values: `60-3600`

Default value: `1800`

Recommended value: `1800`

4.4.19 bgpd.monitor.type

Defines how IRP monitors Improvements to Internet Exchanges:

- by using coarse-grained internal monitors that merely validate an IX peer is live or
- by using fine-grained prefix monitors for each IX improvement that validate that the IX peer still advertises the improved prefix.

⚠ Prefix monitors might consume significant router CPU resources when relying on SNMP to determine if an IX peer advertises the improved prefix and the number of IX improvements is large.

Refer to bgpd.prefix.monitor.interval for details.

Possible values: `0 (Use internal monitor), 1 (Use prefix monitor)`

Default value: `0`

4.4.20 bgpd.no_export

Controls how Bgpd appends NO_EXPORT or NO_ADVERTIZE communities to outbound improvements.

A router is responsible (as defined in RFC 1997) for preventing the utter route redistribution (NO_ADVERTISE) or the route advertisement outside of an autonomous system (NO_EXPORT).

Its strictly recommended to have this option turned on as an additional measure in preventing Outbound Improvement leakage outside of an optimized network.

One may consider disabling the feature in the following conditions:

- Different autonomous system numbers are used inside of an optimized network.
  The feature may prevent redistribution of a Global improvement in MRD configuration.
- Downstream BGP peering.
  The feature may prevent redistribution of routes to downstream, making the improved routes not visible to it.

⚠ Disable with caution

0 Disabled

1 Append NO_EXPORT community to outbound improvements

2 Append NO_ADVERTISE community to outbound imrovements

Possible values: `0, 1, 2`

Default value: `1`

4.4.21 bgpd.policy.cascade.amount

Defines the maximum number of downstream AS for cascading policies. If IRP identifies more downstream elements in AS-PATH from the designated AS the policy for those AS will not be enforced.

Possible values: `1-1000000`

Default value: `1000`

Recommended value: `1000`

4.4.22 bgpd.prefix.monitor.interval

Prefix monitor tracks at given interval in seconds if a peering partner on an Exchange is still announcing the prefix that IRP improved through it.

ⓘ This is intended to avoid cases when after IRP makes an Improvement through a peer on an IX the peer stops announcing/servicing the route.

Possible values: `1-60`

Default value: `10`

4.4.23 bgpd.prefix.monitor.search_interval

Defines the interval between retries of prefix monitor failed initialization attempts in case a prefix isn't advertized by an IX peering partner or if a SNMP error occurs.

During this time IRP will not be able to make improvements through the affected IX peers.

Possible values: `300-3600`

Default value: `300`

4.4.24 bgpd.prefixlist.asn

Specifies a collection of AS numbers that are analyzed for inbound optimization of transiting traffic. Refer Optimization of transiting traffic, Optimization of transiting traffic.

IRP ignores prefixes /25 and shorter for IPv4 and /65 and shorter for IPv6 being present in network's routing table. This means that traffic belonging to these small prefixes are accounted under the immediately larger prefix that fits the above criteria.

Possible values: `list of valid AS numbers`

4.4.25 bgpd.prefixlist.prefixes

Specifies a collection of IPv4 or/and IPv6 prefixes in CIDR notation that are analyzed for inbound optimization of transiting traffic. Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `list of valid CIDR prefixes`

4.4.26 bgpd.rd_local_mark

Specifies a marker to distinguish local improvements from global improvements in the case of multiple routing domains optimization. Parameter represents a valid value for BGP community attribute of the form X:Y. Value in bgpd.rd_local_mark is APPENDED to communities attribute. Refer global.rd_rtt, peer.X.rd, peer.X.flow_agents, rd.X.community.local.

Possible values: `X:Y`

Default value: `65535:1`

4.4.27 bgpd.retry_probing.new.bmp_path_change

Defines on what new provider AS Path changes to re-probe an already improved prefix. The possible options are:

- 0: Disabled
- 1: On major AS-Path change
- 2: On any AS-Path change

Major AS Path changes are those traversing a different set of Autonomous Systems. AS Path changes such as prepended ASN are ignored when only major AS Path changes are monitored.

Refer also to bgpd.retry_probing.old.bmp_path_change, peer.X.bmp.

Possible values: `0, 1, 2`

Default value: `0`

4.4.28 bgpd.retry_probing.old.bmp_path_change

Defines on what old provider AS Path changes to re-probe an already improved prefix. The possible options are:

- 0: Disabled
- 1: On major AS-Path change
- 2: On any AS-Path change

Major AS Path changes are those traversing a different set of Autonomous Systems. AS Path changes such as prepended ASN are ignored when only major AS Path changes are monitored.

Refer also to bgpd.retry_probing.new.bmp_path_change, peer.X.bmp.

Possible values: `0, 1, 2`

Default value: `0`

4.4.29 bgpd.scaninterval

The interval in seconds between the execution of the BGP Scan process.
BGP scan is used for:

- checking the improvements belonging to aggregated routes
- checking the improvements for **aggregate withdrawn(4.4.10)** and for **aggregate next-hop equal to improvements next-hop(4.4.9)** conditions
- checking the improvements for changed BGP attributes
- checking for changes for the improvements to be announced to/withdrawn from iBGP neighbors

Possible values: `10-600` its not recommended to set value higher than 60 seconds

Default value: `20`

4.4.30 bgpd.snmp.packets_interval

Time interval in miliseconds between transit improvement's monitor SNMP packets. Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `1-100`

Default value: `10`

4.4.31 bgpd.snmp.simultaneous

Defines the maximum number of OIDs that can be requested in a single PDU.

Possible values: `1-300`

Default value: `10`

Recommended value: `10`

4.4.32 bgpd.transit.communities

The list of BGP communities that will be appended to an Inbound Transit improvement.

4.4.33 bgpd.transit.monitor.election_interval

Transit monitor election interval as a factor of reconfirm intervals. Refer bgpd.transit.monitor.fast_reconfirm_interval.

Possible values: `1-10`

Default value: `4`

4.4.34 bgpd.transit.monitor.fast_reconfirm_interval

Transit monitor fast reconfirm interval in seconds. The reconfirm interval sets the periodicity by which transit monitors verify presence of alternative routes from other providers for transit improvements.

Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `1-60`

Default value: `15`

4.4.35 bgpd.transit.monitor.retries

Number of SNMP retries before a timeout of transit improvement's monitor. Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `1-2000`

Default value: `3`

4.4.36 bgpd.transit.monitor.timeout

Timeout in miliseconds of individual SNMP requests used to monitor transit improvements. Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `1-10000`

Default value: `1000`

4.4.37 bgpd.updates.split

Instructs Bgpd to split advertised prefixes in two equal parts (e.g /24 is split into two /25 prefixes).

This parameter should be enabled in order to preserve the original BGP UPDATE attributes received from the corresponding aggregates.

Refer to global.aggregate, global.agg_ipv4_max, global.agg_ipv6_max parameters.

> If this option is enabled, the number of announced prefixes will be twice the core.improvements.max.

If the bgpd.peer.X.updates.limit.max parameter value is established, then the limitation is set on the total amount of announced prefixes, AFTER split. For example, if the value of core.improvements.max is set to 10000 and bgpd.peer.X.updates.limit.max is set to 5000, then the amount of the improvements towards this particular provider is no more than 2500, split onto 5000.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.5 BGP sessions settings

4.5.1 bgpd.peer.X.as

Mandatory for each iBGP session definition.

Parameter changes cause reset of BGP session.

Defines the Autonomous System Number for the iBGP session.

Possible values: `1-4294967295`

4.5.2 bgpd.peer.X.blackholing.ipv4.next_hop

Defines IPv4 address which will be used as next_hop in BGP UPDATE for Blackholing routes sent to this router.
The next-hop address should be known by the router.

Possible values: `IPv4 address`

4.5.3 bgpd.peer.X.blackholing.ipv6.next_hop

Defines IPv6 address which will be used as next_hop in BGP UPDATE for Blackholing routes sent to this router.
The next-hop address should be known by the router.

Possible values: `IPv6 address`

4.5.4 bgpd.peer.X.blackholing.localpref

Defines localpref value used by IRP in BGP UPDATE for Blackholing routes sent to this router.

Avoid collisions of localpref values assigned to IRP both within its configuration and/or on customer's network.

Possible values: `0-4294967295`

Default value: `100`

4.5.5 bgpd.peer.X.cap_4byte_as

Parameter changes cause reset of BGP session.

Defines usage of 16 or 32-bit autonomous system numbers. Capability can be negotiated during session setup or forced to either 16 or 32 bits.

- 1 - Negotiated on OPEN
- 2 - Always 16-bit AS path

- 3 - Always 32-bit AS path

When the router is operating in legacy mode and does not negotiate capabilities but still sends 32-bit AS Path then Bgpd considers this a malformed AS_PATH attribute and disconnects the session. To avoid this the parameter must be set to force use of 32bit AS numbers.

For example: A BGP session with IRP with "disable-capability-negotiation" option configured on Vyatta router.

As a result, the BGP session is established and then teared down with log messages:

Listing 4.1: Error log

```
Jan 29 10:20:40.777965 WARN: BGP session RTR/IPv4 (10.0.0.1 AS 65530)
   Incoming UPDATE error: Invalid elements ignored. Malformed AS_PATH
Jan 29 10:20:40.778021 ERROR: BGP session RTR/IPv4 (10.0.0.1 AS 65530)
   Incoming UPDATE error: malformed AS_PATH xxxxxxx
```

The solution is to set `bgpd.peer.RTR.cap_4byte_as = 3`, then the BGP session succeeds.

Possible values: `1 (Negotiate on OPEN), 2 (Always 16-bit AS path), 3 (Always 32-bit AS path)`

Default value: `1`

4.5.6 bgpd.peer.X.flowspec

Enables/disables FlowSpec capability for BGP session.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.5.7 bgpd.peer.X.flowspec.redirect_type

The parameter defines the protocol dialect used to specify FlowSpec redirect.

Typically, Huawei routers use the IETF dialect, while Cisco routers use the Simpson dialect.

Nokia dialect has been implemented as described in the Unicast Routing Protocols Guide Release 22.2.R1.

Possible values: `0 (IETF), 1 (Simpson), 2 (Nokia)`

Default value: `1`

4.5.8 bgpd.peer.X.inbound.ipv4.next_hop

Defines IPv4 address which will be used as next_hop in BGP UPDATE for inbound improvements/announcements towards this router.

The next-hop address should be known by the router.

An inbound rule can be configured with a rule-specific next-hop. Refer to inbound.rule.X.next_hop.

Possible values: `IPv4 address`

4.5.9 bgpd.peer.X.inbound.ipv6.next_hop

Defines IPv6 address which will be used as next_hop in BGP UPDATE for inbound improvements/announcements towards this router.

The next-hop address should be known by the router.

An inbound rule can be configured with a rule-specific next-hop. Refer to inbound.rule.X.next_hop.

Possible values: `IPv6 address`

4.5.10 bgpd.peer.X.inbound.master_localpref

Defines localpref value used by IRP for inbound improvements for this router.

Assigned localpref value should allow Inbound Improvement to become best route.

> Avoid colisions of localpref values assigned to IRP both within its configuration and/or on customer's network.

Possible values: `0-4294967295`

Default value: `102`

4.5.11 bgpd.peer.X.inbound.slave_localpref

Defines localpref value used by IRP for inbound improvements for this router.

Assigned localpref value should allow Inbound Improvement to become best route.

> Avoid colisions of localpref values assigned to IRP both within its configuration and/or on customer's network.

Possible values: `0-4294967295`

Default value: `101`

4.5.12 bgpd.peer.X.keepalive

Defines the session keepalive time (sec). See RFC1771 for details. Hold time is calculated as keepalive * 3.

Possible values: `1-600`

Default value: `60`

Recommended value: 1/3 holdtime

4.5.13 bgpd.peer.X.listen

Instructs the BGP daemon to listen for incoming sessions. It must be set to 1 to be RFC1771 compliant. It can be set to 0 to resolve specific issues.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

Recommended value: `1`

4.5.14 bgpd.peer.X.master_communities

Defines the BGP Community that will be appended by Bgpd to all advertised prefixes. The format is: "`X:Y`".

> Avoid colisions of communities values assigned to IRP both within its configuration and/or on customer's network.

When failover is enabled (global.failover) slave's BGP Community (bgpd.peer.X.slave_communities) must be configured too.

In case Bgpd receives the full or partial RIB (Routing Information Base), values for: **MED**, **Origin**, **LocalPref** and **Communities** are taken from a less-specific Aggregate route. If values for **MED**, **Origin**, **LocalPref** are set in config, it will override any value from Aggregate route. If value for **Communities** is set in config, it will be appended to communities from Aggregate route.

4.5.15 bgpd.peer.X.master_localpref

Defines the local-preference value for prefixes (improvements) announced by Bgpd.

Avoid colisions of localpref values assigned to IRP both within its configuration and/or on customer's network.

When failover is enabled (global.failover) slave's BGP LocalPref (bgpd.peer.X.slave_localpref) must be configured too.

If failover is enabled, master's LocalPref value must be greater than slave's LocalPref value.

In case Bgpd received the full or partial RIB (Routing Information Base), values for **MED**, **Origin**, **LocalPref** and **Communities** will be taken from less-specific Aggregate route. If values for **MED**, **Origin**, **LocalPref** are set in config, it will override any value from Aggregate route. If value for **Communities** is set in config, it will be appended to communities from Aggregate route.

Possible values: `0-4294967295`

Default value:

4.5.16 bgpd.peer.X.master_our_ip

Mandatory for IPv4 BGP session.

Parameter changes cause reset of BGP session.

Defines IPv4 address of IRP server end of this iBGP session. When failover is enabled (global.failover) slave's IP address (bgpd.peer.X.slave_our_ip) must be configured too.

Possible values: IPv4 address

4.5.17 bgpd.peer.X.master_our_ipv6

Mandatory for IPv6 BGP session.

Parameter changes cause reset of BGP session.

Defines IPv6 address of IRP server end of this iBGP session. When failover is enabled (global.failover) slave's IPv6 address (bgpd.peer.X.slave_our_ipv6) must be configured too.

Possible values: IPv6 address

4.5.18 bgpd.peer.X.master_password

Defines iBGP session's password. When failover is enabled (global.failover) slave's local IPv6 address (bgpd.peer.X.slave_password) must be configured too.

Possible values: `up to 80 alphanumeric characters`

4.5.19 bgpd.peer.X.master_peer_ip

⚠ Mandatory for IPv4 BGP session.

ℹ Parameter changes cause reset of BGP session.

Defines iBGP session's router's IPv4 address.

Possible values: IPv4 address

4.5.20 bgpd.peer.X.master_peer_ipv6

⚠ Mandatory for IPv6 BGP session.

ℹ Parameter changes cause reset of BGP session.

Defines iBGP session's router's IPv6 address.

Possible values: IPv6 address

4.5.21 bgpd.peer.X.master_router_id

⚠ Mandatory for IPv6 BGP session either as standalone master or when failover is enabled and the value should be different from bgpd.peer.X.slave_router_id.

ℹ Parameter changes cause reset of BGP session.

Defines IRP server's router ID. The BGP router ID is used in the BGP algorithm for determining the best path to a destination where the preference is given to the BGP router with the lowest router ID.

Possible values: 4-byte value in the IPv4 address format. Any valid IPv4 address can be used.

4.5.22 bgpd.peer.X.med

Defines the Multi-Exit Discriminator (MED) value for prefixes (improvements) announced by Bgpd.

In case Bgpd received the full or partial RIB (Routing Information Base), values for **MED**, **Origin**, **LocalPref** and **Communities** will be taken from less-specific Aggregate route. If values for **MED**, **Origin**, **LocalPref** are set in config, it will override any value from Aggregate route. If value for **Communities** is set in config, it will be appended to communities from Aggregate route.

Possible values: `0-4294967295`

4.5.23 bgpd.peer.X.origin

Defines the Origin value for prefixes announced by Bgpd.

Possible values: `0 (IGP), 1 (EGP), 2 (INCOMPLETE)`

4.5.24 bgpd.peer.X.shutdown

> Parameter changes cause reset of BGP session.

Defines whether the corresponding iBGP session is active or shutdown.

Possible values: `0 (Active), 1 (Shutdown)`

Default value: `0`

4.5.25 bgpd.peer.X.slave_communities

Defines the BGP Community that will be appended by Bgpd on the slave node of a failover configuration to all advertised prefixes. The format is: "`X:Y`".

See also bgpd.peer.X.master_communities.

4.5.26 bgpd.peer.X.slave_localpref

Defines the local-preference value for prefixes announced by Bgpd.
See also bgpd.peer.X.master_localpref.

Possible values: `0-4294967295`

Default value:

4.5.27 bgpd.peer.X.slave_our_ip

> Mandatory for IPv4 BGP session.

> Parameter changes cause reset of BGP session. Affects only BGP session of slave node in a failover configuration.

Defines IPv4 address of IRP server end of this iBGP session of slave node in a failover configuration.
See also bgpd.peer.X.master_our_ip.

Possible values: IPv4 address

4.5.28 bgpd.peer.X.slave_our_ipv6

> Mandatory for IPv6 BGP session in failover configuration.

Parameter changes cause reset of BGP session. Affects only BGP session of slave node in a failover configuration.

Defines IPv6 address of IRP server end of this iBGP session of slave node in a failover configuration.
See also bgpd.peer.X.master_our_ipv6.

Possible values: IPv6 address

4.5.29 bgpd.peer.X.slave_password

Defines iBGP session's password for slave node in a failover configuration.
See also bgpd.peer.X.master_password.

Possible values: `up to 80 alphanumeric characters`

4.5.30 bgpd.peer.X.slave_peer_ip

Optionally distinct router IPv4 address may be specified in the parameter to be used in order to establish the BGP session from slave instance.

Mandatory for IPv4 BGP session.

Parameter changes cause reset of BGP session.

Defines iBGP session's router's IPv4 address.

See also bgpd.peer.X.master_peer_ip.

Possible values: IPv4 address

4.5.31 bgpd.peer.X.slave_peer_ipv6

Optionally distinct router IPv6 address may be specified in the parameter to be used in order to establish the BGP session from slave instance.

Mandatory for IPv6 BGP session.

Parameter changes cause reset of BGP session.

Defines iBGP session's router's IPv6 address.

See also bgpd.peer.X.master_peer_ipv6.

Possible values: IPv6 address

4.5.32 bgpd.peer.X.slave_router_id

Mandatory for IPv6 BGP session either as standalone slave or when failover is enabled and the value should be different from bgpd.peer.X.master_router_id.

Parameter changes cause reset of BGP session. Affects only BGP session of slave node in a failover configuration.

Defines IRP server's router ID. The BGP router ID is used in the BGP algorithm for determining the best path to a destination where the preference is given to the BGP router with the lowest router ID.

Possible values: 4-byte value in the IPv4 address format. Any valid IPv4 address can be used.

4.5.33 bgpd.peer.X.transit.mib

Parameter changes cause reset of BGP session.

Defines the MIB used by transit improvements monitors. Refer Optimization of transiting traffic, Optimization of transiting traffic.

- 0 - Generic (BGP4-MIB)

Possible values: `0`

Default value: `0`

4.5.34 bgpd.peer.X.transit.snmp

Parameter changes cause reset of BGP session.

Points to the SNMP host (and its parameters) used for transit improvements monitor on this session. Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `valid SNMP host identifier`

4.5.35 bgpd.peer.X.transit.status

Parameter changes cause reset of BGP session.

Enables or disables transit improvements through this BGP session (router). Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.5.36 bgpd.peer.X.updates.limit.max

Represents a maximum number of prefixes that can be announced simultaneously in one session.

If bgpd.updates.split is ON, the number of announced prefixes is twice the number of the improvements. If the BGP neighbor (typically - the edge router) has any hardware/software limitation for the number of routes in active routing table, then Bgpd can be instructed not to announce more than a specified amount of prefixes. Value 0 means no limit for current iBGP session.

Values less than maximum allowed improvements, can cause not all the improved prefixes to be injected into such peer.

Higher values can be incompatible with the router (please consult the router's vendor regarding the maximum amount of entries in routing table as well as the BGP table).

Possible values: `0-100000`

Default value: `0`

Recommended value: `0`

4.5.37 bgpd.peer.X.updates.limit.ps

Defines the maximum number of updates per second that will be sent to the current BGP neighbor. Low values will slow down the improvements injection. High values can cause router to drop the improvement without installing it into the routing database.

Possible values: `1-1000000`

Default value: `500`

4.6 BMP monitoring station settings

4.6.1 irpbmpd.grpc.port

Specifies the TCP port on which the Irpbmpd is listening for interprocess communication.

Possible values: `1-65535`

Default value: `7603`

4.6.2 irpbmpd.log

Defines the file-system path to the BMP monitoring station (irpbmpd) log file.

Default value: `/var/log/irp/irpbmpd.log`

4.6.3 irpbmpd.log.level

Defines the logging level for the BMP monitoring station (irpbmpd) service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.6.4 irpbmpd.port

Defines the TCP port where BMP monitoring station (irpbmpd) listens for monitoring routers to establish BMP sessions.

> When BMP information is available for all providers also consider setting BMP as a source of AS_PATH information under bgpd.as_path.

Possible values: `1-65535`

Default value: `7854`

4.6.5 irpbmpd.sources

Defines IP addresses if the valid BMP sourcese. BMP connections coming in from other IP addresses that are not listed in this parameter will be ignored.
It is recommended to specify only trusted IP addresses.

Possible values: `list of valid IPv4/IPv6 addresses`

Default value: `0.0.0.0/0 ::/0`

Recommended value: `Trusted IPv4/IPv6 flow exporter addresses`

4.7 Collector settings

4.7.1 collector.detect.explorer_ips

Instructs the collector to ignore the Explorer-generated traffic, which can initiate false IRP reaction to the network events (Excessive loss, blackout).

This feature is used only by the SPAN collector, instructing it to ignore the IPv4 traffic when the network address translation is used to masquerade IP addresses.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

Recommended value: `0`

4.7.2 collector.export.archive_inbound

Enable Inbound traffic distribution history.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.7.3 collector.export.archive_transit

Enable Transit traffic distribution history.
Warning: may significantly increase disk consumption.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.7.4 collector.export.top_volume_ips

Defines the maximum number of top hosts per prefix for which usage values are stored. Collector persists up to this number of IP address, usage tuples together with other prefix statistics being collected. When the value of the parameter is zero no such statistics will be collected.

Possible values: `0-10`

Default value: `5`

4.7.5 collector.export.ttl

Defines the collector-gathered data lifetime (in seconds). Larger values lead to excessive database size. Aggregated data is kept for one year, disregarding this parameter.

Possible values: `1-1000000`

Default value: `86400`

Recommended value: `1-30days`

4.7.6 collector.export.volume.high.top_n

Defines the number of top volume prefixes.
Top N prefixes will be marked for priority probing in descending order of volume. Lower values lead to small number of events for priority probing of high volume prefixes. Higher values lead to overflow of probing queue with jobs for unimportant prefixes.

Possible values: `0-1000000`

Default value: `50`

Recommended value: `10-50`

4.7.7 collector.export.volume.min

Defines the minimum collected prefix volume (bytes). The prefix will not be exported into the IRP database if its traffic volume is less than the value of the current parameter (collector.export.volume.min) and the percentage of the traffic volume less the collector.export.volume.min_pct parameter.

Lower values will lead to a higher number of prefixes exported into the IRP database.

Possible values: `1-2000000000`

Default value: `100000`

Recommended value: `10000-1000000`

4.7.8 collector.export.volume.min_pct

Defines the minimum prefix volume (%) for a prefix to be exported into the IRP database. The prefix will not be exported into the IRP database if its percentage of the traffic volume is less than the value of the current parameter (collector.export.volume.min_pct) and the traffic volume less than collector.export.volume.min. The percentage of the traffic volume is calculated according to the export interval (60 seconds). Lower values will lead to a higher number of prefixes exported into the IRP database.

Possible values: `0.0001-100`

Default value: `0.01`

Recommended value: `0.01-0.1`

4.7.9 collector.flow.all_outbound

The parameter allows Irpflowd to process all outboind traffic even not listed incollector.ournets.

Enabling the feature requirespeer.X.flow_agents to be properly configured for all providers to distinguish outbound direction.

Possible values: `0 (Ournets only), 1 (All traffic)`

Default value: `0`

4.7.10 collector.flow.buffer.size

Defines the buffer size (in packets) for the Flow collector (irpflowd). Higher values lead to extra memory used by the collector. Slightly increase this value if the network has traffic spikes that cause buffer overflows and packet drop in the collector.

Possible values: `10000-1000000`

Default value: `50000`

Recommended value: `50000-500000`

4.7.11 collector.flow.enabled

Enables the Flow collector. If the parameter is enabled, Flow Collector will be used to gather network prefix data, but it is still possible to use SPAN Collector to acquire network events.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

Recommended value: `0` - for SPAN collector (see collector.span.enabled), `1` - for Flow collector.

4.7.12 collector.flow.export.inbound_transit.topn

Specifies the number of largest volume transit prefixes that are exported each collector cycle for inbound optimization of transiting traffic. Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `1-10000`

Default value: `100`

4.7.13 collector.flow.listen.nf

Defines the UDP port the Flow collector will listen to, for NetFlow traffic.

Possible values: `0-65535` 0 disables listening for NetFlow traffic

Default value: `2055`

4.7.14 collector.flow.listen.sf

Defines the UDP port the Flow collector will listen to, for sFlow traffic.

Possible values: `0-65535` 0 disables listening for sFlow traffic

Default value: `6343`

4.7.15 collector.flow.log

Defines the file-system path to the irpflowd log file.

Default value: `/var/log/irp/irpflowd.log`

4.7.16 collector.flow.log.level

Defines the logging level for the irpflowd service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.7.17 collector.flow.process_transit_in_outbound

Enables or disables matching of prefix traffic at network egress for inbound optimization of transiting traffic. Refer Optimization of transiting traffic, Optimization of transiting traffic.

⚠ This should be enabled only in complex network topologies where ingress prefix statistics can be incomplete.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.7.18 collector.flow.sources

Defines the valid NetFlow/sFlow/jFlow exporters IP addresses. Flow data coming in from other IP addresses that are not listed in this parameter will be ignored.
It is recommended to specify only trusted Flow exporters IP addresses.

Possible values: `list of valid IPv4/IPv6 addresses`

Default value: `0.0.0.0/0 ::/0`

Recommended value: `Trusted IPv4/IPv6 flow exporter addresses`

4.7.19 collector.flow.tcp_ports.limit

TCP port number cap.

Parameter defines the number of TCP ports collected per each outbound IP address.

See also: collector.flow.tcp_ports.mode.

> The parameter is used by Irpflowd to limit the number of ports collected within a one minute interval.
>
> It is also used by Explorer to limit the number of TCP ports utilized for sending probes to a particular IP address.

Possible values: `1-50`

Default value: `5`

4.7.20 collector.flow.tcp_ports.list

TCP ports list.

Defines a list of monitored TCP ports used by an outbound IP address, which are collected when collector.flow.tcp_ports.mode is set to "Collect ports in list".

Possible values: `list of TCP port numbers (1-65535)`

Default value:

See also: collector.flow.tcp_ports.mode.

4.7.21 collector.flow.tcp_ports.max

TCP port range end.

Defines the end of the monitored TCP port range used by an outbound IP address, collected when collector.flow.tcp_ports.mode is set to "Collect ports in range".

Possible values: `1-65535`

Default value: `3000`

See also: collector.flow.tcp_ports.mode.

4.7.22 collector.flow.tcp_ports.min

TCP port range start.

Defines the start of the monitored TCP port range used by an outbound IP address, collected when collector.flow.tcp_ports.mode is set to "Collect ports in range".

Possible values: `1-65535`

Default value: `1`

See also: collector.flow.tcp_ports.mode.

4.7.23 collector.flow.tcp_ports.mode

TCP port collection mode.

Defines the collection mode of TCP ports, used by an outbound IP address.

Collected ports are treated as service ports open for remote connections and available for measuring performance metrics using TCP CONNECT.

0 Do not collect Disables TCP port collection

1 Collect ports in range Remote TCP ports within the range between collector.flow.tcp_ports.min and collector.flow.tcp_ports.max

2 Collect ports in list Remote TCP port present in collector.flow.tcp_ports.list

3 Lesser port number Select the outbound port number if it is smaller than the number chosen by the inbound IP address.

Possible values: `0, 1, 2, 3`

Default value: `0`

See also: explorer.interval.tcp_syn.

4.7.24 collector.ournets

> ⚠ Mandatory ifcollector.span.enabled is enabled

Defines the list of networks to be analyzed. Typically this is the list of prefixes advertised by your AS.

Format:

1. Space-separated list of local IPv4/IPv6 prefixes that should be analyzed and optimized by the IRP.
2. Absolute path to a newline-separated file containing a list of local IPv4/IPv6 prefixes that should be analyzed and optimized by the IRP.

IPv4 prefixes are treated as /24 subnets and IPv6 prefixes as /48 subnets, if netmask is not clearly specified.

> ⓘ The downstream clients' networks can be indicated as well.

Possible values: `IPv4 or IPv6 prefixes`

4.7.25 collector.sessions.max

Defines the maximum number of TCP sessions inside the collector process. It can be used to minimize memory usage.

> ⓘ Both Flow and Span collectors use this parameter.

Estimated memory usage can be calculated as follows: usage = 80MB + (collector.sessions.max*1464)

Possible values: `10000-10000000`

Default value: `2000000`

Recommended value: Depends on available server memory and the maximum number of simultaneous sessions during peak hours.

4.7.26 collector.snmp.enhanced.sec

Defines during what second of a minute the SNMP enhanced algorithm should run.

Possible values: `30-58`

Default value: `45`

Recommended value: `45`

4.7.27 collector.span.buffer.size

Defines the buffer size (in packets) for the Span collector (irpspand). Higher values lead to extra memory used by the collector. Slightly increase this value if the network has traffic spikes that cause buffer overflows and packet drop in the collector.

Possible values: `10000-5000000`

Default value: `100000`

Recommended value: `100000-5000000`

4.7.28 collector.span.enabled

Enables the Span collector. Span collector will be used to gather network prefix data, if the parameter is enabled.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

Recommended value: `0` - if no traffic analysis should be performed by the Span collector, `1` - if the Span collector should analyze mirrored traffic for network events and/or prefix statistics.

See also: collector.flow.enabled

4.7.29 collector.span.interfaces

⚠ Mandatory ifcollector.span.enabled is enabled

Defines a space-separated network interfaces list for passive packet analysis by the Span collector.
Example:

```
collector.span.interfaces = eth0 eth1 eth2
```

4.7.30 collector.span.log

Defines the file-system path to the irpspand log file.

Default value: `/var/log/irp/irpspand.log`

4.7.31 collector.span.log.level

Defines the logging level for the irpspand service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.7.32 collector.span.min_delay

Enables fast probing tasks queuing for prefixes with one of the following issues:

- Blackouts
- Congestion
- Excessive packet delays.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

Recommended value: `0`

See also: collector.span.min_delay.probing_queue_size

4.7.33 collector.span.min_delay.probing_queue_size

Defines the number of slots in the probing queue that can be used by the min_delay algorithm (see: collector.span.min_delay)

Possible values: `1-200`

Default value: `50`

Recommended value: `30-200`

See also: collector.span.min_delay

4.7.34 collector.span.size_from_ip_header

When parameter is enabled packet size is determined from header of the IPv4/IPv6 packet. Otherwise, packet size is determined from link layer information (original packet size from PCAP/SNF).

⚠ Enable this when source packets are stripped before entering Noction IRP's appliance SPAN interface.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.7.35 collector.span.threshold.blackout

Defines the percentage of retransmitted packets in regards to the total packets number. If retransmit is higher than this value, this prefix is considered to have a blackout.

Possible values: `1-100`

Default value: `70`

Recommended value: `70-90`

4.7.36 collector.span.threshold.congestion

Defines the percentage of retransmitted packets in regards to the total packets number. If retransmit is higher than this value, this prefix is considered to have a congestion.

Possible values: `0-100`

Default value: `50`

Recommended value: `30-70`

4.7.37 collector.span.threshold.delay

Percentage of excessive delayed packets by the total packets number (see collector.span.threshold.excessive). If this percentage is higher than this value, we consider this prefix to have a delay.

Possible values: `1-100`

Default value: `20`

Recommended value: `10-30`

4.7.38 collector.span.threshold.excessive

Defines the excessive RTT (%). The value is compared to the average round trip time. If RTT is higher by collector.span.threshold.excessive in % than the average RTT, then the counter of excessive delay packets is incremented.

Possible values: `100-500`

Default value: `200`

Recommended value: `100-500`

4.7.39 collector.speaking_ips

Defines the number of speaking IPs to be stored in the database.

Possible values: `0-1000`

Default value: `100`

Recommended value: `100`

4.8 Core settings

4.8.1 core.circuit.high_loss_diff

Defines the upper threshold of consistent packet loss over a provider when compared to other providers on the network. A provider exceeding this threshold is marked for shutdown. An event of this type will be raised and available to subscribers to act upon. Threshold loss difference is determined over a configured past time horizon and compared with all other providers on the network over the same interval.

⚠ While the provider is marked for shutdown IRP cannot do this itself. Instead network engineers and/or other network capabilities are expected to be triggered in order to divert as much traffic as possible away from provider with circuit issues.

Possible values: `2-50`

Default value: `15`

4.8.2 core.circuit.hist_interval

Defines time interval in minutes used to determine average packet loss over a provider when compared to other providers on the network for circuit issues detection algorithm.

⚠ Keep in mind that shorter time intervals might cause false positives where the averages are high simply because a few large and random packet loss probes are able to push the numbers above thresholds. At the same time longer time intervals will take longer to spot issues and thus will extend the time period where a circuit with issues is being used while alternatives were available.

Possible values: `1-240`

Default value: `5`

4.8.3 core.circuit.inbound

Defines if and when IRP announces Max prepends for inbound prefixes through provider with circuit issues. The options are as follows:

- No changes - any prepends through provider are not changed
- Prepend on warn - IRP announces Max Prepends once Warning level for circuit issues is reached
- Prepend on shutdown - IRP announces Max Prepends only when shutdown level for circuit issues is exceeded.

Refer also core.circuit.transit.

Possible values: `0 (No changes), 1 (Prepend on warn), 2 (Prepend on shutdown)`

Default value: `0`

4.8.4 core.circuit.recover_hold_time

Defines the time interval in seconds before IRP attempts to restore a circuit with issues.

Possible values: `60-3600`

Default value: `600`

4.8.5 core.circuit.recover_loss_diff

Defines the normal threshold of packet loss when a provider with circuit issues can be considered to be ok. At this stage IRP will restore the provider to its full capacity status and will retract all other measures taken in the past in order for the network traffic to avoid the circuit with issues.

This parameter is only used if the ways to react to a circuit issue include restoring it and this only can happen within the given time interval. Otherwise the circuit should be restored by manual intervention of network engineers after they have verified the issue is no longer a problem.

Note that this parameter should be both smaller than core.circuit.high_loss_diff and core.circuit.warn_loss_diff

Possible values: `0-48`

Default value: `5`

4.8.6 core.circuit.recover_monitored_intervals

Defines the time interval in minutes during which IRP will continuously evaluate average loss for provider(s) with circuit issues in order to determine if it is back to normal.

This parameter is only used if the ways to react to a circuit issue include restoring it and this only can happen within the given time interval. Otherwise the circuit should be restored by manual intervention of network engineers after the circuit issue is no longer a problem.

Possible values: `1-30`

Default value: `5`

4.8.7 core.circuit.transit

Defines if and when IRP announces Max prepends for transit prefixes through provider with circuit issues. The options are as follows:

- No changes - any prepends through provider are not changed
- Prepend on warn - IRP announces Max Prepends once Warning level for circuit issues is reached
- Prepend on shutdown - IRP announces Max Prepends only when shutdown level for circuit issues is exceeded.

Refer also core.circuit.inbound.

Possible values: `0 (No changes), 1 (Prepend on warn), 2 (Prepend on shutdown)`

Default value: `0`

4.8.8 core.circuit.warn_loss_diff

Defines the lower threshold of packet loss when a provider seems to be having circuit issues. At this stage IRP will start raising alerts that network engineers and/or network management systems can subscribe in order to act on them. At this level IRP starts taking other preventive measures such as re-probing outbound improvements made to provider with circuit issues.

Note that this parameter should be both smaller than core.circuit.high_loss_diff and larger than core.circuit.recover_loss_diff.

Possible values: `1-49`

Default value: `10`

4.8.9 core.circuit.withdraw_on_warn

Enables or disables withdrawing outbound improvements to a provider with circuit issues at or above warning level.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.8.10 core.commit_control

Enables or disables Commit Control (see the Commit Control section)

The following parameters must be set in the configuration file to ensure proper functionality of the Commit Control feature:

1. SNMP parameters must be set for each provider:
   (a) SNMP Host
   (b) peer.X.snmp.interfaces
2. Each provider needs to have peer.X.95th set to the desired Commit level
3. peer.X.precedence must be set for each provider

When Commit Control is disabled, all existing Commit Control improvements are removed from current improvements. The improvements are preserved temporarily until core.commit_control.probe_ttl after Core service restart. If Commit Control is re-enabled before expiration these improvements will be restored into current improvements.
See also:
Commit Control
core.commit_control.rate.group
core.commit_control.rate.high
core.commit_control.rate.low

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.8.11 core.commit_control.agg_bw_min

Defines the minimum bandwidth for a single prefix. Prefixes, whose bandwidth is less than the specified value, are ignored by Commit Control and are not being used in the Commit Control algorithm.

It is not recommended to set high values for this parameter since it limits the number of prefixes that can be rerouted by the Commit Control mechanism.

This parameter is considered only during initial improvement. During retry probing of existing Commit Control improvements IRP might detect that the current bandwidth usage for some prefixes is below this limit. IRP will preserve these improvements as relevant if there are no other reasons to withdraw them.

Its recommended to decrease value if summary throughput is lower than 200-500 Mbps.

Possible values: `0.1-5000`

Default value: `1`

Recommended value: `1`

4.8.12 core.commit_control.del_irrelevant

If enabled, Commit Control algorithm deletes improvements that have traffic volume less than value in the4.8.11 parameter.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

Recommended value: `1`

4.8.13 core.commit_control.group_balance.check_monitor

Defines how monitors affect group load balancing.
Group load balance will include providers in a group only when a provider has no failed monitors.

0 Ignores monitors

1 Excludes providers with failed IPv4 monitor

2 Excludes providers with any failed monitor

Possible values: `0, 1, 2`

Default value: `0`

4.8.14 core.commit_control.inbound.enabled

Enables or disables Inbound bandwidth control.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.8.15 core.commit_control.inbound.improvement.delay

The parameter specifies minimal time delay before next Inbound/Transit optimization cycle.
Time delay should be sufficient to cover route propagation time and time required to collect and process changes in bandwidth distribution.

Possible values: `60-1800`

Default value: `300`

4.8.16 core.commit_control.inbound.moderated

Enables or disables review and moderate feature of inbound bandwidth control.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.8.17 core.commit_control.inbound.rate.high

Defines provider's high load rate limit (%). Refer core.commit_control.rate.high for details.

This limit is used when inbound and outbound commit control operate independently. Otherwise core.commit_control.rate.high value overrides this. Refer also peer.X.95th.mode.

Possible values: `50-99`

Default value: `90`

4.8.18 core.commit_control.inbound.rate.low

Defines provider's low load rate limit (%). Refer core.commit_control.rate.low for details.

This limit is used when inbound and outbound commit control operate independently. Otherwise core.commit_control.rate.low value overrides this. Refer also peer.X.95th.mode.

Possible values: `30-99`

Default value: `70`

4.8.19 core.commit_control.inbound.sync_groups

Synchronous Inbound Provider Groups allows for the Inbound Commit Control decisions to be applied simultaneously to all the providers within the group. The number of applied prepends for a single provider would propagate to all the providers in the group.

Typical use case: When providers are in the same autonomous system, prepending prefixes via a single provider leads to a complete traffic shift to another one, potentially overloading the link. Arrange all the providers within a given ASN into a group to apply the same amount of prepends for each provider.

A group consists of provider IDs separated by double colons. Groups in the parameter should be separated by spaces.

⚠ Any number of providers can be included in a group, but a provider cannot belong to more than one group.

4.8.20 core.commit_control.inbound.volume_estimation

Defines method of estimating inbound bandwidth. The available options are:

- 0: Last minute data
- 1: Largest of last minute and current hour average
- 2: Largest of last minute and daily average

Possible values: `0, 1, 2`

Default value: `1`

4.8.21 core.commit_control.loss_override

Defines when IRP allows Commit Control improvements to adjust a provider's bandwidth considering current and new provider's loss measurements. The available options are to allow Commit Control improvements when there is:

- 0: Better or equal loss
- 1: Better or irrelevant loss difference
- 2: Any loss difference
- 3 (Allow for unsuccessful probing)

Refer core.performance.loss_pct for details.

Possible values: `0, 1, 2, 3`

Default value: `0`

4.8.22 core.commit_control.probe_ttl

Defines the TTL (time-to-live) for a specific probe. If the probe results are older than the value specified here, the system will disregard it and the Commit Control algorithm will schedule it for expedited re-probing.

Possible values: `600-86400`

Default value: `7200`

Recommended value: `7200`

4.8.23 core.commit_control.probing_queue_size

Defines the number of slots in the probing queue that can be used by the Commit Control algorithm. This sets the upper limit on the number of prefixes that can be scheduled for probing by Commit Control. Note that this queue has higher priority than ordinary and retry probing. At the same time Commit Control relies heavily on existing probing results and most of the time this queue will be empty.

Possible values: `1-500`

Default value: `100`

Recommended value: `10-100`

4.8.24 core.commit_control.rate.group

If using provider load balancing in group, this parameter defines allowed deviation of a provider's current bandwidth(in %) compared with other providers in the same group.

Example 1. There are three grouped providers with 95th set to 1 Gbps, 2 Gbps and 3 Gbps with a total current bandwidth of 600 Mbps. In this case, load balancing expects that each provider's bandwidth usage will be 100 Mbps, 200 Mbps and 300 Mbps accordingly. These values are proportional to individual provider's 95th settings in the group. Let's assume core.commit_control.rate.group is set to 5%. If a provider's bandwidth exceeds by more than the 5% limit the expected value (105 Mbps, 210 Mbps and 315 Mbps accordingly and the total for the group did not change), IRP will start active rerouting of excessive bandwidth from that provider to providers within or outside this group.

> Load balancing in a provider group is performed even when the 95th is not exceeded.

Possible values: `1-30`

Default value: `5`

Recommended value: `5`

See also: Provider load balancing

4.8.25 core.commit_control.rate.high

Defines provider's high load rate limit (%).
IRP will stop Latency/Cost improvement if provider bandwidth is over core.commit_control.rate.high % of load (percents of 95th). The improvements will start happening again after provider's bandwidth drops below core.commit_control.rate.low % of load.

These parameters are used for passive 95th overload prevention as well as an additional method for Commit Control to be less aggressive.

> Provider's Latency/Cost improvement ability does not change when current bandwidth rate varies between core.commit_control.rate.low and core.commit_control.rate.high.

Example 2. if provider's current load is 1100 Mbps, 95th is set to 1000 Mbps. Commit Control will try to unload 200Mbps (to reduce current load to 90% of the 95th level) in order to prevent 95th overload as well as allow provider's bandwidth natural growth during peak hours.

See also: Provider load balancing

Possible values: `50-99`, but larger or equal to core.commit_control.rate.low

Default value: `90`

Recommended value: `90`

4.8.26 core.commit_control.rate.low

Defines provider's low load rate limit (%).
IRP will start Latency/Cost improvements again after provider's bandwidth drops below core.commit_control.rate.low % of load.

Latency/Cost improvements will be stopped if provider bandwidth is over core.commit_control.rate.high % of load (percents of 95th).

These parameters are used for passive 95th overload prevention as well as an additional method for Commit Control to be less aggressive.

> Provider's Latency/Cost improvement ability does not change when current bandwidth rate varies between core.commit_control.rate.low and core.commit_control.rate.high.

See also: Provider load balancing

Possible values: `30-99`, but lower or equal to core.commit_control.rate.high

Default value: `80`

Recommended value: `80`

4.8.27 core.commit_control.react_on_collector

Defines if Commit Control algorithm should react immediately on collector overload values.

Enabling this feature represents a tradeoff. Take into consideration the benefits of a faster react time vs reliance on older probes and statistics when making Commit Control improvements.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

Recommended value: `1`

4.8.28 core.commit_control.worst_loss

Defines amount of loss worsening allowed when IRP performs Commit Control improvements and core.commit_control.loss_overrideis set to “Any loss difference”.

Refer core.performance.loss_pct for details.

Possible values: `1-99`

Default value: `2`

4.8.29 core.cost.worst_ms

Latency worsening.

Defines the allowed latency worsening for an improved prefix while running in Cost optimization mode (see global.improve_mode) and the Commit Control feature is enabled. While operating in these modes IRP will consider as alternatives candidates with latency degradation not exceeding this limit. Of course, the candidate with least latency degradation or even with better latency will be chosen as an improvement.

When IRP detects that an existing Commit Control improvement has alternatives with latencies better than specified value it will replace it with a new performance (latency) improvement. Over-usage of the alternative route will be avoided.

This parameter is applicable for local improvements within a Routing Domain. Global improvements will also take into account core.global.worst_ms values.

Possible values: `1-1000000`

Default value: `10`

Recommended value: `10`

4.8.30 core.eventqueuelimit

Defines the exploring events queue size for IPv4 prefixes.

Lower values may result in IRP missing some important network issues.

Possible values: `1-10000`

Default value: `1000`

Recommended value: `500-2000`

4.8.31 core.eventqueuelimit.retry_probe_pct

Defines a percentage of the total exploring queue length (see core.eventqueuelimit) to be used for retry probing.

Possible values: `1-100`

Default value: `40`

Recommended value: `20-60`

4.8.32 core.eventqueuelimit_ipv6

Defines the exploring events queue size for IPv6 prefixes.
Lower values may result in IRP missing some important network issues.

Possible values: `1-10000`

Default value: `1000`

Recommended value: `500-2000`

4.8.33 core.flowspec.max

Defines the maximum number of IPv4 Flowspec rules that IRP is allowed to advertise.

> ⚠ Routers have a low limit of FlowSpec entries. If the limit is exceeded, the router behavior may be unpredictable.

Possible values: `1-20000`

Default value: `100`

4.8.34 core.flowspec.max_ipv6

Defines the maximum number of IPv6 Flowspec rules that IRP is allowed to advertise.

> ⚠ Routers have a low limit of FlowSpec entries. If the limit is exceeded, the router behavior may be unpredictable.

Possible values: `1-20000`

Default value: `100`

4.8.35 core.global.allow_commit

Enables or disables global commit control Improvements in a multiple routing domain configuration. By default these Improvements are enabled.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.8.36 core.global.allow_latency_cost

Enables or disables latency and cost global Improvements in a multiple routing domain configuration. By default these Improvements are enabled.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.8.37 core.global.worst_ms

Defines acceptable latency worsening for cost and commit for global improvements.

Refer core.cost.worst_ms

Possible values: `1-1000000`

Default value: `30`

Recommended value: `10`

4.8.38 core.improvements.clearslots

Specifies the number of outdated improvements that will be discarded when slots are needed for new improvements. Outdated are improvements as defined by core.improvements.clearslots.days_max. In case there are no outdated improvements usual improvement replacement mechanisms are employed relying on Improvements weight.

Possible values: `0-100`

Default value: `10`

Recommended value: `10`

4.8.39 core.improvements.clearslots.days_max

Sets the number of days after which an improvement is considered outdated. Outdated improvements have not been re-confirmed for a long time. The oldest of them will be discarded by the slot clearing process for outdated improvements when new slots are needed for new improvements. Refer core.improvements.clearslots.

Possible values: `1-60`

Default value: `7`

Recommended value: `7`

4.8.40 core.improvements.inbound_transit.max

Specifies the maximum number of transit improvements. Refer Optimization of transiting traffic, Optimization of transiting traffic.

> ⚠ Increasing this limit should be done with care. Each transit improvement is continuously monitored if alternative routes from providers are still present on the router(s) and this consumes router CPU resources.

Possible values: `1-1000`

Default value: `100`

4.8.41 core.improvements.inbound_transit.ttl.clean

Specifies how IRP should treat old transit improvements.
Old transit improvements are those that exceed core.improvements.inbound_transit.ttl.max.
The options are to either

- gradually decrease the number of prepends and withdraw the transit improvement when no prepends are left or
- withdraw the inbound transit improvement immediately when it exceeds core.improvements.inbound_transit.ttl.max.

Transit improvements cleanup performed once a day at configured off-peak hour (global.offpeak_hour).

Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `0 (Decrease prepends), 1 (Remove immediately)`

Default value: `0`

4.8.42 core.improvements.inbound_transit.ttl.max

Specifies the maximum period of time to preserve a transit improvement. If IRP does not have any other reasons to adjust a transit improvement after this limitation is exceeded its prepends will be decreased or the improvement will be withdrawn. Refercore.improvements.inbound_transit.ttl.clean, Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `901-345600`

Default value: `86400`

4.8.43 core.improvements.inbound_transit.ttl.min

Specifies the minimal period of time to preserve a transit improvement. In order to give sufficient time to the entire Internet to adjust to a transit improvement and to avoid route instability IRP blocks making changes to a transit improvement for this duration of time. IRP can still offer route changes to the same transit prefixes routed through other providers. Refer Optimization of transiting traffic, Optimization of transiting traffic.

Possible values: `600-86400`

Default value: `14400`

4.8.44 core.improvements.max

Defines the maximum number of active IPv4 improvements.

The number of announced prefixes can be twice this value if bgpd.updates.split is enabled.

Possible values: `1-100000`

Default value: `10000`

Recommended value: `10000`

4.8.45 core.improvements.max_ipv6

Defines the maximum number of active IPv6 improvements.

Possible values: `0-100000`

Default value: `2000`

Recommended value: `2000`

4.8.46 core.improvements.safe_removal

Specifies when IRP removes/withdraws outbound improvements. Outbound improvements become irrelevant if old provider performance metrics are no longer worse than the improved provider metrics. Still, original routes for some improvements change and providers that will service those prefixes once IRP improvements are withdrawn might have worse performance metrics. Safe improvement removal accounts for this possibility and removes improvements only when ALL providers have performance metrics that are not worse than the (fresh) improvement provider metrics.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.8.47 core.improvements.ttl.retry_probe

Defines the retry probing interval (in seconds). Once an improvement is older than this value, it will be sent for re-probing.

Possible values: `600-86400`

Default value: `14400`

Recommended value: `7200-14400`

4.8.48 core.log

Defines the file-system path to the core log file.

Default value: `/var/log/irp/core.log`

4.8.49 core.log.level

Defines the logging level for the core service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.8.50 core.outage_detection

Enables the Outage Detection algorithm.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

Recommended value: `1`

4.8.51 core.outage_detection.limit_pct

Defines the problem prefixes threshold (in percent) over which the system confirms an outage (if core.outage_detection is enabled).

See also: Outage detection

Possible values: `0-100`

Default value: `60`

4.8.52 core.outage_detection.volume_top_n

Top-N relevant volume prefixes

The outage detection mechanism considers only those prefixes in the "Top N by volume." Separately for IPv4 and IPv6.

Possible values: `1-20000`

Default value: `5000`

4.8.53 core.overusage.check_interval

Defines the interval to check overusage for Flowspec policies prefixes in order to apply automatic throttling Flowspec policies.

Possible values: `60-600`

Default value: `60`

Recommended value: `60`

4.8.54 core.overusage.hold_timer

Defines the retention time for throttling rules after prefix average bandwidth returns to normal for automatic throttling Flowspec policies.

Possible values: `60-600`

Default value: `60`

Recommended value: `300`

4.8.55 core.overusage.out.average.period

Defines the duration of the time interval in hours used to determine average prefix usage for automatic throttling Flowspec policies.

Possible values: `1-24`

Default value: `1`

Recommended value: `1`

4.8.56 core.overusage.out.average.relevant_min

Defines minimum prefix bandwidth average (in Mbps) that is relevant for automatic throttling Flowspec policies.

Possible values: `1-100000`

Default value: `50`

Recommended value: `50`

4.8.57 core.overusage.out.threshold.throttle

Defines the multiplier of prefix average bandwidth applied on outbound traffic for automatic throttling Flowspec policies.

Possible values: `2-10000`

Default value: `2`

Recommended value: `2`

4.8.58 core.overusage.out.threshold.trigger

Defines the multiplier of prefix average bandwidth that sets the overusage threshold of a prefix. An automatic throttling Flowspec policy is created when current prefix bandwidth exceeds the average multipled by this multiplier.

Possible values: `2-10000`

Default value: `10`

Recommended value: `10`

4.8.59 core.performance.loss_pct

Defines the relevant packet loss difference (in percent) after which a loss improvement is made.

For current route loss > 50%: Do not perform rerouting, if packet loss difference between routes is less than 15%.

For current route loss <= 50%: Do not perform rerouting, if packet loss difference between routes is less than core.performance.loss_pct.

Possible values: `1-15`

Default value: `3`

Recommended value: `3-5`

4.8.60 core.performance.rtt.diff_ms

Defines the relevant RTT difference (ms) after which a latency improvement is made. If the RTT difference between the current route and another provider is less than core.performance.rtt.diff_ms, then the system ignores this prefix as an improvement candidate or removes the current improvement as irrelevant during the improvement reconfirmation process.

> core.performance.rtt.diff_ms and core.performance.rtt.diff_pct are grouped together at decision making, using a logical AND condition.

Possible values: `1-1000000`

Default value: `10`

Recommended value: `2-20`

4.8.61 core.performance.rtt.diff_pct

Defines the relevant RTT difference (in percent) after which a latency improvement is made. If the RTT difference between the current route and another provider is less than core.performance.rtt.diff_pct, then the system ignores this prefix as an improvement candidate or removes the current improvement as irrelevant during the improvement reconfirmation process.

> core.performance.rtt.diff_ms and core.performance.rtt.diff_pct are grouped together at decision making, using a logical AND condition.

Possible values: `1-100`

Default value: `10`

Recommended value: `5-20`

4.8.62 core.performance.rtt.ix_diff_ms

Defines the relevant RTT difference (ms) to make latency improvements across Internet Exchanges and transit providers. A latency improvement from an IX to a transit provider is allowed only when the latency is improved by at least this threshold. Inversely, a latency improvement from a transit provider to an IX is allowed even if latency degradation is less thanthis threshold.

> A default value of zero for this parameter disables this feature. A value smaller than or equal to core.performance.rtt.diff_ms isn't allowed. When the feature is disabled then the core.performance.rtt.diff_ms and core.performance.rtt.diff_pct parameters are considered to determine relevancy of latency improvements.

> core.performance.rtt.ix_diff_ms and core.performance.rtt.ix_diff_pct are grouped together at decision making, using a logical AND condition.

Possible values: `0-1000000`

Default value: `0`

Recommended value: `20-50`

4.8.63 core.performance.rtt.ix_diff_pct

Defines the relevant RTT difference (in percent) to make latency improvements across Internet Exchanges and transit providers. Refer core.performance.rtt.ix_diff_ms for details.

> core.performance.rtt.ix_diff_ms and core.performance.rtt.ix_diff_pct are grouped together at decision making, using a logical AND condition.

Possible values: `1-100`

Default value: `10`

Recommended value: `5-20`

4.8.64 core.probes.ttl.failed

Defines the failed probe lifetime (in seconds).

Regular and Retry probing will not be performed until the age of the failed probe is less than core.probes.ttl.failed

Possible values: `120-14400`

Default value: `300`

Recommended value: `300`

4.8.65 core.probes.ttl.max

Defines the probe lifetime (in seconds).

Probed prefixes data is kept in the IRP database for the specified amount of time. Automatic cleanup of outdated data is performed periodically by Dbcron.

See also: Administrative Components

High values will lead to database size growth.

Possible values: `86400-604800`

Default value: `86400`

Recommended value: `86400`

4.8.66 core.probes.ttl.min

Defines the relevant probe lifetime (in seconds)

Ordinary probing will not be performed for a specific prefix if its probe age is less than core.probes.ttl.min.

Possible values: `300-43200`

Default value: `7200`

Recommended value: `1800-14400`

4.8.67 core.problem.outage_timeout

Outage detection timeout (in seconds).

The IRP will disregard a possible outage, if it was not confirmed during last core.problem.outage_timeout.

Possible values: `60-3600`

Default value: `600`

See also: core.outage_detection

4.8.68 core.problem.rtt.diff_pct

Defines the RTT difference (in percent) between the current route and the new route, for Outage detection algorithm. IRP will choose a new route for problematic prefixes only if the RTT difference (in percent) is greater than this parameter.

Possible values: `1-100`

Default value: `50`

Recommended value: `30-60`

4.8.69 core.vip.interval.probe

Defines the VIP prefixes probing interval, in seconds.

All networks and ASNs specified in `/etc/noction/policies.conf` and will be queued for priority probing every core.vip.interval.probe seconds.

> For probing intervals lower than 5 minutes (300 seconds) IRP uses fast probing algorithms avoiding for example long lasting tracing.

See also: VIP Improvements

Possible values: `60-14400`

Default value: `3600`

4.9 Explorer settings

4.9.1 explorer.aipi

If enabled, AIPI (Adaptive Inter-Packet Interval) algorithm is executed using inter-packet intervals configured for each provider.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

Recommended value: `0`

See also: peer.X.diag_hop.interval_min, peer.X.diag_hop.interval_max

4.9.2 explorer.algorithms

Enables or disables the use of new scanning, probing and tracing Explorer algorithms.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.9.3 explorer.high_vol_precedence

High volume tasks have priority over the tasks with variable cardinality.

The parameter establishes the balance between the tasks allocated for processing high volume prefixes (high priority for the Customer network) and the network events tasks (such as blackout, congestion, etc).

Possible values: `10-90`

Default value: `50`

See also: explorer.maxthreads, collector.export.volume.high.top_n

4.9.4 explorer.infra_ips

Defines the IPv4/IPv6 addressees, used in internal infrastructure to determine the current route for a specific prefix. If netmask is not specified, then /32 is assumed for IPv4 address and /128 for IPv6 address (host addresses).

Format:

1. Space-separated list of infrastructure IPv4/IPv6 addresses.
2. Absolute path to a newline-separated file containing a list of local IPv4/IPv6 prefixes that should be analyzed and optimized by the IRP.

Possible values: See above.

Recommended value: all IPv4/IPv6 addresses used in the internal infrastructure.

4.9.5 explorer.interval.infra

Defines the time interval (in milliseconds) between the packets sent to infrastructure hops (4.9.4). By decreasing this value the Explorer performance enhances.

Possible values: `1-200`

Default value: `5`

Recommended value: `5`

4.9.6 explorer.interval.other

Defines the time interval (in milliseconds) between the packets sent to destination hops (4.9.4).

Possible values: `1-1000`

Default value: `200`

Recommended value: `200`

4.9.7 explorer.interval.other.trace

Defines the time interval (in milliseconds) between traceroute packets sent to non-infrastructure hops.

Possible values: `1-1000`

Default value: `20`

Recommended value: `20`

4.9.8 explorer.interval.tcp_syn

TCP packets interval (ms).

Defines the interval (in milliseconds) between TCP probes sent to a particular IP address.

Possible values: `1-1000`

Default value: `10`

See also: collector.flow.tcp_ports.mode.

4.9.9 explorer.ipv4_test

Defines public IPv4 address that will be used for ensuring that PBR is operational.

Possible values: `IPv4 address`

Default value: `8.8.8.8`

Recommended value: `8.8.8.8`

4.9.10 explorer.ipv6_test

Defines public IPv6 address that will be used to verify that PBR is operational.

Possible values: `IPv6 address`

Default value: `2620:0:ccc::2`

Recommended value: `2620:0:ccc::2`

4.9.11 explorer.log

Defines the file-system path to the explorer log file.

Default value: `/var/log/irp/explorer.log`

4.9.12 explorer.log.level

Defines the logging level for the explorer service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.9.13 explorer.max_collector_ips

Process max collected IPs. Maximum number of IPs to probe for a specific prefix.

Possible values: `1-256`

Default value: `10`

4.9.14 explorer.maxthreads

Defines the number of simultaneously processed exploring tasks.

> ⚠ If this value is excessively large Explorer tasks can take very long to finish. Hardware/router diagnostic packet rate limitations kick in causing all threads to wait for exploring tasks to finish.

Possible values: `10-2000`

Default value: `200`

Recommended value: `200-300`

4.9.15 explorer.probe.algorithm

Defines the probing algorithm(s) used by explorer, in the preferred order.
The following algorithms can be used:

- UDP - udp requests (destination port: 33434)
- ICMP - regular ICMP ping sweep
- TCP_SYN - tcp syn scan (refer to: TCP port collection for outbound IPs)

Possible values: `ICMP UDP TCP_SYN`

Default value: `ICMP UDP TCP_SYN`

Recommended value: `ICMP UDP TCP_SYN`

4.9.16 explorer.probe.indirect_priority

Defines the execution priority between direct and indirect probing algorithms. Parameter value of 1 means that indirect probing will be executed prior to direct probing algorithm. Parameter value of 2 disables indirect probing.

> This parameter conflicts with explorer.trace.all configuration option.

Possible values: `0 (Direct then Indirect), 1 (Indirect then Direct), 2 (Direct only), 3 (Direct plus Outage tracing)`

Default value: `0`

Recommended value: `0`

4.9.17 explorer.probing.sendpkts.adaptive_max

Defines the maximum number of ping packets for adaptive prefix probing.

Possible values: positive Integer value, higher or equal to explorer.probing.sendpkts.min

Possible values: `1-1000`

Default value: `100`

4.9.18 explorer.probing.sendpkts.min

Defines the default ping packets count for each probe. If any loss is detected, additional packets (up to explorer.probing.sendpkts.adaptive_max) are sent, for a more accurate packet loss detection.

Possible values: `1-1000`

Default value: `5`

4.9.19 explorer.probing.simultaneous

Defines the number of scanned IP addresses.

Possible values: `1-100`

Default value: `50`

Recommended value: `50`

4.9.20 explorer.scanning.confirm_ips

Defines the number of scanned speaking IP addresses for a provider with loss.

Possible values: `1-10`

Default value: `5`

Recommended value: `5`

4.9.21 explorer.scanning.replypkts.min

Defines the number of response packets from a scanned speaking IP address to qualify as a candidate.

Possible values: `1-10`

Default value: `5`

Recommended value: `5`

4.9.22 explorer.scanning.rtt.dispersion_ms

Defines RTT maximum dispersion in miliseconds to qualify a speaking IP as candidate.

Possible values: `1-1000`

Default value: `50`

Recommended value: `50`

4.9.23 explorer.scanning.sendpkts.factor

Defines the number of attempts to send packets to all selected speaking IPs.

Possible values: `1-10`

Default value: `5`

Recommended value: `5`

4.9.24 explorer.timeout

Defines the probes ICMP timeout (in ms)

Possible values: `50-10000`

Default value: `2000`

4.9.25 explorer.timeout.infra

Defines the waiting time (in milliseconds) for infrastructure hops' (4.9.4) responses expected during trace execution.

Possible values: `50-20000`

Default value: `10000`

Recommended value: `10000`

4.9.26 explorer.trace.algorithms

Defines the traceroute algorithm(s) to be used, arranged by priority. See explorer.probe.algorithm for algorithms description.

Possible values: `UDP ICMP TCP_SYN`

Default value: `UDP ICMP`

Recommended value: `UDP ICMP`

4.9.27 explorer.trace.all

Defines tracing behaviour. If traces are forced, each probed prefix unconditionally runs traces through all configured providers. Regular tracing is needed for Outage Detection and for AS Path reconstruction. Traces can be disabled altogether by setting explorer.trace.all to 2.

⛔ Third algorithm should be enabled in bgpd.as_path when traces are fully disabled.

⚠ Forcing ALL traces (explorer.trace.all = 1) can significantly slow down probing for networks. Review probing times before and after changing this parameter and if probing times become unacceptable revert to previous setting.

ℹ When all traces are disablled (explorer.trace.all = 2) IRP is missing essential trace data and is not able to take action on some types of problems. This effectively cuts off those features.

Possible values: `0 (Regular), 1 (Force all), 2 (Disable all)`

Default value: `0`

Recommended value: `0`

4.9.28 explorer.trace.diag_hop_unique

Enforces the uniqueness of per-provider diagnostic hops peer.X.ipv4.diag_hop, peer.X.ipv6.diag_hop.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.9.29 explorer.traceroute.retrypkts

Defines the number of additional traceroute packets to be sent to the intermediate hop in case it has not replied and might be an infrastructure boundary hop.

Possible values: `0-50`

Default value: `10`

Recommended value: `10`

4.9.30 explorer.traceroute.sendpkts

Defines the number of traceroute packets per hop to be sent for each probe.

Possible values: `3-5`

Default value: `3`

Recommended value: `3`

4.9.31 explorer.traceroute.simultaneous

Defines the number of simultaneously probed hops during trace execution initiated from the hop defined by (4.9.34).

Possible values: `1-30`

Default value: `5`

Recommended value: `5`

4.9.32 explorer.traceroute.simultaneous.infra

Defines the number of simultaneously probed infrastructure hops during trace execution.

Possible values: `1-30`

Default value: `3`

Recommended value: `3`

4.9.33 explorer.traceroute.ttl.max

Defines the maximum traceroute TTL. Essentially this defines the last hop at which explorer stops the trace.

Possible values: `1-255`

Default value: `30`

Recommended value: `30`

4.9.34 explorer.traceroute.ttl.min

Defines the minimum traceroute TTL. Basically this defines the first hop after which explorer analyzes the trace.

Possible values: `1-255`

Default value: `2`

Recommended value: `2-5`

4.10 Inbound Performance

Feature overview: Inbound performance optimization
How to configure: Inbound performance optimization configuration

4.10.1 irpinperfd.enabled

Enables or disables the inbound performance feature.
Refer to: 1.2.18

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.10.2 irpinperfd.log

Defines the path to the log file in the server file-system.

Default value: `/var/log/irp/irpinperfd.log`

4.10.3 irpinperfd.log.level

Defines the logging level for the Irpinperfd service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

4.10.4 irpinperfd.model.debug

Enables additional logging of Inbound Performance statistical model.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.10.5 irpinperfd.model.shelf_life

The value indicates the time interval (in seconds) for which the calculated model will be used, after which the model will be recalculated.

Possible values: `7220-43200`

Default value: `7220`

4.10.6 irpinperfd.model.stability_interval

Time interval (in seconds) for algorithm to gather data to evaluate how many prefixes from the model have recorded traffic.

Possible values: `1800-3600`

Default value: `1800`

4.10.7 irpinperfd.model.topn_per_rule

The number of top volume prefixes to be analyzed per an inbound performance rule.
Increasing this value may raise the accuracy of improvements at the expense of the probing slots.

Possible values: `1-1000`

Default value: `200`

4.10.8 irpinperfd.moderated

Enables or disables the moderated mode of handling inbound improvements.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.10.9 irpinperfd.probing.confirmation_interval

The time interval (seconds) between two probings of the sample prefixes for active rules.

Possible values: `3600-86400`

Default value: `18000`

4.10.10 irpinperfd.probing.failure_margin

The value indicates the minimal acceptable percentage of the failed probes.

Possible values: `0-25`

Default value: `10`

4.10.11 irpinperfd.probing.interval

Time interval (seconds) between two probings of the sample prefixes.

Possible values: `300-5400`

Default value: `600`

4.10.12 irpinperfd.probing.shelf_life

The value indicates how much time (seconds) the probe will be actual for inbound performance improvement.

Possible values: `30-3600`

Default value: `600`

4.10.13 irpinperfd.probing.timeout

Defines the waiting time (seconds) of the probing results. Probing result absence will lead to the algorithm restarts if probing is not finished within the configured time period.

Possible values: `120-900`

Default value: `600`

4.10.14 irpinperfd.rtt_allowed_worsening

Defines the maximum allowed worsening of RTT per probe.
After rule activation, if a significant number of probes will experience worsening of RTT greater than this value, the improvement will be removed.

Possible values: `1-1000`

Default value: `10`

4.11 Threat mitigation

4.11.1 irpdetectd.bgp.reaction

Threat Mitigation BGP blackholing reaction that gets used by default when no custom rule is defined.

Possible values: `0 (Drop), 1 (Redirect)`

Default value: `0`

4.11.2 irpdetectd.bgp.redirect.bgp_peers

The list of BGP communities for BGP redirect default reaction.
The list of BGP router(s) that receive BGP redirect announcements.

> The parameter is also used by the Bgpd component.

Possible values: `list of BGP routers`

4.11.3 irpdetectd.bgp.redirect.communities

> The parameter is also used by the Bgpd component.

Possible values: `list of BGP communities`

4.11.4 irpdetectd.blackhole.threshold.kpps

Blackhole threshold kpps.
Default kilo packets per second limit that triggers a blackholing event.
A value of 0 disables the feature functionality.
The default rate can be overridden by a custom rule.

Possible values: `0-1000000`

Default value: `0`

4.11.5 irpdetectd.blackhole.threshold.mbps

Blackhole threshold mbps.
Default megabits per second limit that triggers a blackholing event.
A value of 0 disables the feature functionality.
The default rate can be overridden by a custom rule.

Possible values: `0-1000000`

Default value: `0`

4.11.6 irpdetectd.flowspec.ipv4.redirect

The default IPv4 address used by Threat Mitigation FlowSpec redirect reaction.

> The parameter is also used by the Bgpd component.

Possible values: `IPv4 address`

4.11.7 irpdetectd.flowspec.ipv6.redirect

The default IPv6 address used by Threat Mitigation FlowSpec redirect reaction.

The parameter is also used by the Bgpd component.

Possible values: `IPv6 address`

4.11.8 irpdetectd.flowspec.reaction

Threat Mitigation FlowSpec reaction that gets used by default when no custom rule is defined.

Possible values: `0 (Drop), 1 (Redirect)`

Default value: `0`

4.11.9 irpdetectd.flowspec.threshold.kpps

Flowspec threshold kpps.
Default kilo packets per second limit that triggers a Flowspec event.
A value of 0 disables the feature functionality.
The default rate can be overridden by a custom rule.

Possible values: `0-1000000`

Default value: `0`

4.11.10 irpdetectd.flowspec.threshold.mbps

Flowspec threshold mbps.
Default megabits per second limit that triggers a Flowspec event.
A value of 0 disables the feature functionality.
The default rate can be overridden by a custom rule.

Possible values: `0-1000000`

Default value: `0`

4.11.11 irpdetectd.ipv4.prefix_size

The default size of an IPv4 prefix which gets blocked by the BGP/FlowSpec threat mitigation action.

Possible values: `16-32`

Default value: `32`

4.11.12 irpdetectd.ipv6.prefix_size

The default size of an IPv6 prefix which gets blocked by the BGP/FlowSpec threat mitigation action.

Possible values: `32-128`

Default value: `128`

4.11.13 irpdetectd.mode

DDoS Mode.
Threat Mitigation modes:
Automatic - threat mitigation actions are performed automatically when an attack gets detected;
Moderated - users need to confirm the threat mitigation action manually;
Disabled - turns off Threat Mitigation altogether.

Possible values: `0 (Disabled), 1 (Manual), 2 (Moderated), 3 (Automated)`

Default value: `1`

4.11.14 irpdetectd.protected_addresses

Protected addresses.

Possible values: `0 (Protect analyzed prefixes), 1 (Protect all prefixes)`

Default value: `0`

4.11.15 irpdetectd.time.keep

Time keep.
The amount of time (minutes) to keep an approved/automatic flowspec/blackholing event active.

Possible values: `5-4320`

Default value: `1800`

4.11.16 irpdetectd.time.monitor

Time monitor.
Amount of time (minutes) between the DDoS attack detection and the automatic activation of the defense mechanism.

Possible values: `2-60`

Default value: `3`

4.11.17 irpdetectd.whitelist

Whitelist.
Prefixes that should not be considered for blocking by the DDoS detection mechanism.

Possible values: `IPv4 or IPv6 prefixes`

4.12 Notification and events

4.12.1 pushd.email.auth.password

Defines SMTP server authentication password.

4.12.2 pushd.email.auth.username

Defines SMTP server authentication username.

4.12.3 pushd.email.from

Defines the From email address used for sending email notifications.

Possible values: `valid email address`

Default value: `root@localhost`

4.12.4 pushd.email.host

Defines the IPv4, IPv6 address or email server host name used to send emails.

Possible values: `Hostname or IPv4/IPv6 address`

Default value: `127.0.0.1`

4.12.5 pushd.email.port

Defines the TCP port used for sending emails.

Possible values: `1-65535`

Default value: `25`

Recommended value: `25`

4.12.6 pushd.listen.port

Defines the TCP listen port of irppushd service.

Possible values: `1-65535`

Default value: `10499`

4.12.7 pushd.log

Defines the complete path to the irppushd log file

Possible values: `full path to log file`

Default value: `/var/log/irp/irppushd.log`

4.12.8 pushd.log.level

Defines the logging level for the irppushd service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.12.9 pushd.sms.account_sid

Defines the public account identifier issued to user by the SMS gateway. Usually this is a random string.

Possible values: `random string`

4.12.10 pushd.sms.auth_token

Defines the secret authentication token issued by the SMS gateway to the account holder. Usually this is a longer random string.

Possible values: `random string`

4.12.11 pushd.sms.gateway

Defines the SMS gateway preferred by the user. A valid account with this SMS gateway is required to send SMS.

Possible values: `none, twilio, plivo`

Default value: `none`

4.12.12 pushd.sms.message_size

Defines the maximum size of SMS texts. Texts that are longer than this limit will be trimmed and a ... will be added. The SMS gateway will split the SMS text into multiple messages if the request includes a longer than supported SMS message.

> The SMS gateway enforces its own text size limits. In case the notification exceeds SMS gateway limits the SMS message can be either trimmed or rejected.

Possible values: `150-6000`

Default value: `150`

4.12.13 pushd.sms.phone_number

Defines the From phone number used for sending SMS notifications.

> SMS gateways might have policies regarding the valid phone numbers that they will accept. Check with your SMS gateways if there are any restrictions and if yes what phone numbers can be provided in this configuration parameter.

Possible values: valid phone number

4.12.14 pushd.sms.uri.plivo

Defines the URL of the Plivo SMS API.

> Do not change this parameter unless Plivo SMS API URL has changed.

Possible values: `valid URL`

Default value: `https://api.plivo.com/v1/Account/%1%/Message/`

Recommended value: `https://api.plivo.com/v1/Account/%1%/Message/`

4.12.15 pushd.sms.uri.twilio

Defines the URL of the Twilio SMS API.

Do not change this parameter unless Twilio SMS API URL has changed.

Possible values: `valid URL`

Default value: `https://api.twilio.com/2010-04-01/Accounts/%1%/Messages.json`

Recommended value: `https://api.twilio.com/2010-04-01/Accounts/%1%/Messages.json`

4.12.16 pushd.templates.datadir

Defines the directory for notification templates used by irppushd service to format email and sms messages.

Possible values: `full path to notification templates`

Default value: `/etc/noction/templates`

4.12.17 pushd.webhook.avatar_emoji

Defines an emoji to be used as the IRP bot's avatar image. The emoji is expected to be in the ":robot-face:" notation.

The avatar emoji is used if an avatar icon in pushd.webhook.avatar_urlis not specified.

Possible values: `valid emoji in :colon: notation`

Default value: `:robot-face:`

4.12.18 pushd.webhook.avatar_url

Defines the URL of the IRP bot's avatar image. URL should be a publicly accessible PNG or JPEG image that the webhook provider is able to retrieve and use.

This is the default avatar image. The avatar icon will be used instead of an emoji if both are specified. Refer to pushd.webhook.avatar_emoji.

Possible values: `valid URL`

Default value: `http://www.noction.com/round-logo.png`

Recommended value: `http://www.noction.com/round-logo.png`

4.12.19 pushd.webhook.botname

Defines the name assigned to IRP bot.

Possible values: `text`

Default value: `IRP`

Recommended value: `IRP`

4.12.20 pushd.webhook.url

Defines the URL assigned to Web Hooks user/team. Web Hooks work by POST-ing JSON content to this designated URL for example

```
POST https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX
```

⚠ This parameter is required in order for any subscription to web hooks channels to work. Currently only the Slack.com web hooks API has been tested and confirmed to work.

Possible values: `valid URL`

4.12.21 trap.bgpd_announced_rate_low.limit_pct

Defines the announcements rate limit percentage.

Possible values: `1-100`

Default value: `60`

Recommended value: `60`

4.12.22 trap.core_cc_improvements_spike.diff_pct

Defines the size (in percent) of the spike in the number of improvement compared to preceeding period average defined in trap.core_cc_improvements_spike.period_sec.

Possible values: `1-100`

Default value: `50`

4.12.23 trap.core_cc_improvements_spike.period_sec

Defines the time interval preceeding a spike. The number of improvements is averaged over this time and a spike event is triggered when the subsequent measurement exceeds the size (in percent) defined by trap.core_cc_improvements_spike.diff_pct.

Possible values: `30-86400`

Default value: `300`

4.12.24 trap.core_cc_overload.inbound_limit_mbps

Defines the overall overload limit in Mbps for the Inbound Commit Control overload by X Mbps event. The event is raised if the overall inbound traffic of the network exceeds the configured value (sum of peer.X.95th for all providers) by this limit.

Possible values: `1-100000`

Default value: `100`

4.12.25 trap.core_cc_overload.inbound_limit_pct

Defines the overall overload limit in percent for the Inbound Commit Control overload by X% event. The event is raised if the overall inbound traffic of the network exceeds the configured value (sum of peer.X.95th for all providers) by this limit.

Possible values: `1-1000`

Default value: `10`

4.12.26 trap.core_cc_overload.limit_mbps

Defines the overall overload limit in Mbps for the Outbound Commit Control overload by X Mbps event. The event is raised if the overall outbound traffic of the network exceeds the configured value (sum of peer.X.95th for all providers) by this limit.

Possible values: `1-100000`

Default value: `100`

4.12.27 trap.core_cc_overload.limit_pct

Defines the overall overload limit in percent for the Outbound Commit Control overload by X% event. The event is raised if the overall outbound traffic of the network exceeds the configured value (sum of peer.X.95th for all providers) by this limit.

Possible values: `1-1000`

Default value: `10`

Recommended value: `10`

4.12.28 trap.core_cc_provider_overload.inbound_limit_mbps

Defines the per provider overload limit in Mbps for the Inbound Commit Control provider X overloaded by Y Mbps event. The event is raised if the inbound traffic for a provider exceeds the configured value (refer peer.X.95th) by more than this limit.

Possible values: `1-100000`

Default value: `100`

4.12.29 trap.core_cc_provider_overload.inbound_limit_pct

Defines the per provider overload limit in percent for the Inbound Commit Control provider X overloaded by Y% event. The event is raised if the inbound traffic for a provider exceeds the configured value (refer peer.X.95th) by more than this limit.

Possible values: `1-1000`

Default value: `10`

4.12.30 trap.core_cc_provider_overload.limit_mbps

Defines the per provider overload limit in Mbps for the Outbound Commit Control provider X overloaded by Y Mbps event. The event is raised if the outbound traffic for a provider exceeds the configured value (refer peer.X.95th) by more than this limit.

Possible values: `1-100000`

Default value: `100`

4.12.31 trap.core_cc_provider_overload.limit_pct

Defines the per provider overload limit in percent for the Outbound Commit Control provider X overloaded by Y% event. The event is raised if the outbound traffic for a provider exceeds the configured value (refer peer.X.95th) by more than this limit.

Possible values: `1-1000`

Default value: `10`

Recommended value: `10`

4.12.32 trap.core_excessive.prefixes

Defines list of prefixes to restrict events prefixExcessiveLatency & prefixExcessiveLoss.
Events would be generated if the list is empty or aggregate from the list contains a prefix from event.

Possible values: `list of valid IPv4/IPv6 prefixes`

Default value: `300`

Recommended value: `300`

4.12.33 trap.core_excessive_latency.limit

Defines the packet latency limit in milliseconds.

Possible values: `1-10000`

Default value: `300`

Recommended value: `300`

4.12.34 trap.core_excessive_loss.limit

Defines the packet loss percentage limit.

Possible values: `1-100`

Default value: `50`

Recommended value: `50`

4.12.35 trap.destination.auth_password

SNMP user's password for IRP generated traps. Applicable only when SNMP v3 with authentication is used. Refer to trap.destination.version, trap.destination.seclevel, trap.destination.auth_username.

Possible values: `text`

4.12.36 trap.destination.auth_protocol

SNMP authentication protocol for IRP generated traps. Applicable only when SNMP v3 with authentication is used. Refer to trap.destination.version, trap.destination.seclevel.

- 0 - MD5
- 1 - SHA

Possible values: `0, 1`

Default value: `1`

4.12.37 trap.destination.auth_username

SNMP user name for IRP generated traps. Applicable only when SNMP v3 with authentication is used. Refer to trap.destination.version, trap.destination.seclevel.

Possible values: `text`

4.12.38 trap.destination.community

Defines the SNMP community used for sending SNMP traps.

Possible values: textual SNMP v2c community name

Default value: `public`

4.12.39 trap.destination.port

Defines the UDP port used for sending SNMP traps.

Possible values: `1-65535`

Default value: `162`

4.12.40 trap.destination.priv_password

SNMP password for IRP generated traps. Applicable only when SNMP v3 with privacy is used. Refer to trap.destination.version, trap.destination.seclevel.

Possible values: `text`

4.12.41 trap.destination.priv_protocol

SNMP encryption protocol for IRP generated traps. Applicable only when SNMP v3 with privacy is used. Refer to trap.destination.version, trap.destination.seclevel.

DES isn't supported in RedHat Enterprise Linux 9

- 0 - DES
- 1 - AES

Possible values: `0, 1`

Default value: `1`

4.12.42 trap.destination.seclevel

SNMP security services for IRP generated traps.
IRP supports either No security, Authentication only or both Authentication and Privacy.
Applicable only when SNMP v3 is used (trap.destination.version).
Depending on security services used further parameters should be configured, for example:
trap.destination.auth_username, trap.destination.auth_password, trap.destination.auth_protocol, trap.destination.priv_password, trap.destination.priv_protocol.

- 0 - Neither Authentication nor Privacy
- 1 - Authentication only
- 2 - Authentication and Privacy

Possible values: `0, 1, 2`

Default value: `0`

4.12.43 trap.destination.version

SNMP version for IRP generated traps.

- 2 - SNMP v2c
- 3 - SNMP v3

Possible values: `2, 3`

Default value: `2`

4.13 Administrative settings

4.13.1 dbcron.api.log

Defines file-system path to the dbcron API log file.

Default value: `/var/log/irp/api.mlog`

4.13.2 dbcron.api.socket

Defines dbcron API listen socket.

Default value: `/var/spool/irp/api`

4.13.3 dbcron.log

Defines the file-system path to the dbcron log file.

Default value: `/var/log/irp/dbcron.log`

4.13.4 dbcron.log.level

Defines the logging level for the dbcron service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.13.5 globalcc.key_service.port

Specifies the TCP port used by the Globalcc component for the purpose of public key exchange.

Possible values: `1-65535`

Default value: `7600`

4.13.6 globalcc.log

Defines the file-system path to the Globalcc log file.

Default value: `/var/log/irp/globalcc.log`

4.13.7 globalcc.log.level

Defines the logging level for the Globalcc service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.13.8 globalcc.service.port

Specifies the TCP port on which the Irpmng is listening for interprocess communication.

Possible values: `1-65535`

Default value: `7601`

4.13.9 irpdetectd.log

Defines the file-system path to the Irpdetectd log file.

Default value: `/var/log/irp/irpdetectd.log`

4.13.10 irpdetectd.log.level

Defines the logging level for the Irpdetectd service.

Possible values: `emerg, fatal, alert, crit, error, warn, notice, info, debug, trace`

Default value: `info`

Recommended value: `info`

4.13.11 irpmng.grpc.port

Specifies the TCP port on which the Irpmng is listening for interprocess communication.

Possible values: `1-65535`

Default value: `7604`

4.14 Provider

4.14.1 peer.X.95th

> Mandatory if core.commit_control is enabled.

Defines current provider's desired 95th value (also known as Committed Interface Rate).

The value of peer.X.limit_load(if set to positive value) must be greater or equal to value of peer.X.95th.

Please refer to core.commit_control and Commit Control for Commit Control description.

Depending on the peer.X.95th.bill_day parameter value, the behavior of the 95th level calculation has two different states.

If peer.X.95th.bill_day is set to `-1`, then this parameter is treated as "committed interface rate limit" and actual 95th calculation is not performed. In this case, IRP will actively reroute traffic from an overloaded provider to keep it below the peer.X.95th level.

If peer.X.95th.bill_day is set to a positive value, then the system calculates the actual 95th value based on the historical traffic information from peer.X.95th.bill_day to the current date of this month. Then, the result is compared to the peer.X.95th value, and the maximum of these two is chosen as the target 95th. Thus, if the 95th for the current month has already exceeded the peer.X.95th value, IRP will keep the traffic usage from increasing above current 95th.

Possible values: `1-9999999`

4.14.2 peer.X.95th.bill_day

Defines the first billing period day. If current month has less days than the specified value, then last day of the month will be taken as the start of the new billing period. The parameter is used for 95th percentile calculation.

If `-1` is specified, then peer.X.95th is treated as "Committed Interface Rate".

Possible values: `-1, 1-31`

Default value: `1`

Recommended value: First day of the billing period.

4.14.3 peer.X.95th.centile

Defines the actual percentage for the 95th percentile, as some providers may have non-standard traffic billing policies.

Possible values: `1-99`

Default value: `95`

Recommended value: please consult your agreements with this specific provider

4.14.4 peer.X.95th.in

Defines the 95th level (in Mbps) for inbound when inbound and outbound commit control operate independently. Refer peer.X.95th, peer.X.95th.mode.

Possible values: `1-9999999`

4.14.5 peer.X.95th.mode

Defines the way how 95th value is determined. The following modes are supported:

- 0: Separate 95th for in/out
- 2: 95th from greater of in or out
- 3: Largest of the two 95th for in/out

Only mode with separate 95th for each inbound and outbound traffic keeps inbound and outbound commit control independent of each other.

Refer also peer.X.95th, peer.X.95th.in.

Possible values: `0, 2, 3`

Default value: `2`

4.14.6 peer.X.aspath_for_ix

Modifies the way IX AS number is treated for outbound improvements towards an Internet Exchange provider for 2nd and 3rd options of bgpd.as_path. The supported options are:

- 0: Strip IX ASN if present
- 1: Preserve IX ASN if present

Refer also bgpd.as_path.

Possible values: `0, 1`

Default value: `0`

4.14.7 peer.X.auto_config

Enables or disables periodic execution of Internet Exchanges auto re-configuration process once per 4.2.6 interval.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.14.8 peer.X.bgp_peer

⚠ Mandatory

Defines the iBGP session(s) over which an improvement made for this provider is advertised. This parameter should be set to a valid BGP neighbor name as defined in BGP sessions settings. Typically, this is the session with the Edge router, which this provider is connected to. Multiple sessions should be set only if there are some additional (backup) links to this provider on multiple Edge routers.

Possible values: One or a list of valid BGP neighbor names, as defined in BGP sessions settings.

4.14.9 peer.X.blackholing.bgp_peer

Specifies a BGP session(s) to be used to send blackholing announcements to.

Possible values: One or a list of valid BGP neighbor names, as defined in BGP sessions settings.

4.14.10 peer.X.blackholing.community

Specifies a BGP community mark to be used in order to distinguish routes that are to be sent towards the provider's router responsible for blackhole routes.

4.14.11 peer.X.bmp

Defines BMP data usage for this provider.

- 0: Do not use BMP data
- 1: Use BMP data if available
- 2: Use BMP data only

Possible values: `0, 1, 2`

Default value: `0`

4.14.12 peer.X.bmp.check_routes

Allows using BMP to check route availability from a Provider configured as a partial routing or IX.

⚠ IRP may unexpectedly use routes received from a Provider, but filtered by an edge router, because BGP filters aren't applied to BMP data.
The parameter must be enabled before switching peer.X.bmp to "Use BMP data only" mode.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.14.13 peer.X.cc_disable

Instructs IRP to exclude this provider from the Commit Control algorithm.

⚠ Has effect only if core.commit_control is enabled.

Possible values: `1 (CC Disabled), 0 (CC Enabled)`

Default value: `0`

4.14.14 peer.X.circuit.control

Defines whether provider will be monitored for excessive loss issues and how to react to them. The possible values are:

- 0: Disabled
- 1: Warn only
- 2: Warn and disconnect
- 3: Disconnect and restore

⚠ IRP tries to induce a disconnect of BGP session via FlowSpec rules that drop any further packets on the BGP session with provider's router. It is thus recommended that the keepalive timer is set to 20 seconds for BGP sessions with providers that are expected to be disconnected if circuit issues are detected on them. This will minimize the time a circuit with excessive loss is kept operational and causes harm.

Possible values: `0, 1, 2, 3`

Default value: `0`

4.14.15 peer.X.cost

Defines the cost per Mbps for the current provider. All peer.X.cost parameters should be specified in the same currency.

ℹ Parameter's value should be the same for each provider in a provider's group (refer to 4.14.56) and cost mode is enabled (4.2.25)

Possible values: `0-1000000000`

Default value: `0`

4.14.16 peer.X.description

Defines the provider's description (full provider name)

Possible values: text

4.14.17 peer.X.diag_hop.interval_max

Defines maximum value for Adaptive Inter-Packet Interval. Adaptive Inter-Packet Interval is used while sending packets to a specific Provider's router (packets with a specific TTL). Inter-packet interval is raised up to the maximum value when the router does not respond to the packets and is decreased to the minimum one in case the router responds. The Adaptive Inter-Packet Interval is changed gradually to obtain better performance.

A value of the 4.9.5 parameter is used in case zero value is specified in this parameter.

Possible values: `0-50000000` between 0ms and 50ms

Default value: `0`

4.14.18 peer.X.diag_hop.interval_min

Defines minimum value for Adaptive Inter-Packet Interval. Adaptive Inter-Packet Interval is used while sending packets to a specific Provider's router (packets with a specific TTL). Inter-packet interval is raised up to the maximum value when the router does not respond to the packets and is decreased to the minimum one in case the router responds. The Adaptive Inter-Packet Interval is changed gradually to obtain better performance.

Possible values: `10000-10000000` between 0.01ms and 10ms

Default value: `1000000`

4.14.19 peer.X.flow_agents

Specifies a collection of Flow agents in the form IPv4/interfaceIdentifier. Flow agents are used to assign specific Flow statistics to a designated provider. Refer Flow agents, Optimization for multiple Routing Domains for details.

Possible values: forward slash separated IP address and numeric identifier in 1..2147483647 range

4.14.20 peer.X.flowspec.ipv4.redirect_community

Defines the BGP Community that will be appended by Bgpd to advertised Flowspec redirect policies for IPv4 sessions. The format is: "`X:Y`".

Avoid collisions of communities values assigned to IRP both within its configuration and/or on customer's network.

4.14.21 peer.X.flowspec.ipv6.redirect_community

Defines the BGP Community that will be appended by Bgpd to advertised Flowspec redirect policies for IPv6 sessions. The format is: "`X:Y`".

Avoid collisions of communities values assigned to IRP both within its configuration and/or on customer's network.

4.14.22 peer.X.flowspec.pbr.enabled

Enables or disables Flowspec PBR for a provider.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.14.23 peer.X.flowspec.pbr.use_bgp_peer

Specifies whether FlowSpec PBR entries should be advertised only to assigned router(s) or to all routers with FlowSpec PBR enabled.

Possible values: `0 (All routers), 1 (Assigned routers)`

Default value: `0`

4.14.24 peer.X.global_group

Includes the provider into a existing Global Group.

See also: Global Group

Possible values: `1-100`

4.14.25 peer.X.group_loadbalance

Defines whether commit control algorithm load balances the group of providers or optimizes it on aggregated bandwidth usage. For details refer Provider load balancing and Commit control of aggregated groups.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.14.26 peer.X.improve_in_group

Enables or disables improvements within a provider group. This parameter must be equal for all providers in a provider group.

If disabled, IRP will not make any Cost or Performance improvements inside the group.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

Recommended value: depends on network infrastructure and policies

4.14.27 peer.X.inbound.community_base

Defines the base community of consecutive range of eight community values which IRP uses to announce inbound improvements for this provider over BGP. Base community instructs an announcement of a route without any additional prepend via a particular provider. Next communities in the range instructs for additional prepends from one up to seven.

> For example: when peer.1.inbound.community_base is set to 65530:50100, the value 65530:50102 represents “prepend 2 times while advertizing to provider 1”.

Possible values: `valid BGP community attribute`

4.14.28 peer.X.ipv4.diag_hop

Mandatory

Defines the diagnostic hop or subnet in CIDR format for the current provider. Usually, it is the IP address of the first hop of the specific provider. The parameter is used to make sure that a route actually passes through this provider.

It is also used for Internet Exchanges autoconfiguration process. The Peering Partners next-hops will be gathered based on the provided subnet(s) in CIDR format.

Any parameter changes (via Frontend) will caused the Bgpd component restart if the provider type (peer.X.type) is Internet Exchanges

See also: explorer.trace.diag_hop_unique, peer.X.ipv6.diag_hop

Possible values: List of valid IPv4 addresses or prefixes (usually equal to peer.X.ipv4.next_hop)

4.14.29 peer.X.ipv4.master_probing

Mandatory

Defines the local IPv4 address, assigned for probing via the current provider. When failover is enabled (global.failover) slave's probing IP (peer.X.ipv4.slave_probing) must be configured too.

See also: Specific PBR configuration scenarios

Possible values: valid local IPv4 address

4.14.30 peer.X.ipv4.mon

Defines the List of IPv4 addresses to be monitored by Bgpd to ensure that the immediate upstream path through this provider is operational. Each specified IP address is probed every bgpd.mon.keepalive seconds to verify accessibility from Bgpd (ICMP/UDP pings are used). Highly available IP addresses that reliably answer to ICMP/UDP pings should be used. It is remommended to setup at least two IP addresses belonging to different networks.

If all monitored IP addresses do not respond for bgpd.mon.holdtime seconds, then any improvements designated for this provider will be withdrawn from edge router(s). Improvements will be re-announced after all IP addresses respond within bgpd.mon.guardtime seconds.

Possible values: space-separated list of valid IPv4 addresses

Default value: `208.67.222.222 8.8.8.8`

4.14.31 peer.X.ipv4.next_hop

Mandatory

Defines the next-hop IPv4 address for BGP route injection. Usually, it is the IPv4 address of the BGP partner from the provider.

Possible values: valid IPv4 address

4.14.32 peer.X.ipv4.next_hop_as

Defines the provider autonomous system number for IPv4. This parameter must be set in order for the 3rd algorithm of Bgpd AS-PATH to work properly (refer to bgpd.as_path).

If this parameter is set, then it's value will be used as part of AS-PATH for outgoing improvements if the 3rd algorithm is enabled in bgpd.as_path.

In case of Internet Exchanges, the AS-Path begins with peering partner's AS number, instead of AS number of route server.

The first AS number will be stripped from AS-Path when advertising improvement towards Exchange in case the first AS number is equal to value set in this parameter.

See also: provider.X.rule.Y.next_hop_as

Possible values: `0-4294967295`

Default value: `0`

4.14.33 peer.X.ipv4.route_server

Defines the Internet Exchange Route Server(s) IP address(es) used to properly auto-configure provider.X.rule.Y.bgp_peer parameter.

See also peer.X.mon.ipv4.internal.mode, peer.X.mon.ipv4.internal.state

Possible values: valid IPv4 address(es)

4.14.34 peer.X.ipv4.slave_probing

Defines the local IPv4 address, assigned for probing via the current provider for the slave node in a failover configuration.

See also: Specific PBR configuration scenarios

4.14.35 peer.X.ipv4_pbr_check

Defines per-provider alternative IPv4 address to be used for PBR tests. This overrides explorer.ipv4_test.

See also: peer.X.ipv6_pbr_check

Possible values: valid IPv4 address

4.14.36 peer.X.ipv6.diag_hop

⚠ Mandatory

Defines the IPv6 diagnostic hop for the current provider. Usually, it is the IP address of the first hop of the specific provider. The parameter is used to verify that a route actually passes through this provider.

See also: explorer.trace.diag_hop_unique, peer.X.ipv4.diag_hop

Possible values: List of valid IPv6 addresses or prefixes, usually equal to peer.X.ipv6.next_hop

4.14.37 peer.X.ipv6.master_probing

⚠ Mandatory for IPv6

Defines the local IPv6 address assigned for probing via the current provider. When failover is enabled (global.failover) slave's probing IPv6 address (peer.X.ipv4.slave_probing) must be configured too.

See also: Specific PBR configuration scenarios

Possible values: valid local IPv6 address

4.14.38 peer.X.ipv6.mon

Defines the List of IPv6 addresses to be monitored by Bgpd to ensure that the immediate upstream path through this provider is operational. Each specified IP address is probed every bgpd.mon.keepalive seconds to verify accessibility from Bgpd (ICMP/UDP pings are used). Highly available IP addresses that reliably answer to ICMP/UDP pings should be used. It is remommended to setup at least two IP addresses belonging to different networks.

If all IP addresses do not respond for bgpd.mon.holdtime seconds, then any improvements designated for this provider will be withdrawn from the edge router(s). Improvements will be announced again after all IP addresses respond within bgpd.mon.guardtime seconds.

Possible values: space-separated list of valid IPv6 addresses

Default value: `2620:0:ccc::2 2001:4860:4860::8888`

4.14.39 peer.X.ipv6.next_hop

⚠ Mandatory for IPv6

Defines the next-hop IPv6 address for BGP route injection. Usually, it is the IPv6 address of the BGP partner from the provider.

Possible values: valid IPv6 address

4.14.40 peer.X.ipv6.next_hop_as

Defines the provider autonomous system number for IPv6. This parameter must be set in order for the 3rd algorithm of Bgpd AS-PATH to work properly (refer to bgpd.as_path).

If this parameter is set, then it's value will be used as part of an AS-PATH for outgoing improvements if the 3rd algorithm is enabled in bgpd.as_path.

See also: provider.X.rule.Y.next_hop_as

Possible values: `0-4294967295`

Default value: `0`

4.14.41 peer.X.ipv6.route_server

Defines the Internet Exchange Route Server(s) IP address(es) used to properly auto-configure provider.X.rule.Y.bgp_peer parameter.

See also peer.X.mon.ipv6.internal.mode, peer.X.mon.ipv6.internal.state

Possible values: valid IPv6 address(es)

4.14.42 peer.X.ipv6.slave_probing

Defines the local IPv6 address assigned for probing via the current provider for the slave node in a failover configuration.

See also peer.X.ipv6.master_probing.

Possible values: valid local IPv6 address

4.14.43 peer.X.ipv6_pbr_check

Defines per-provider alternative IPv6 address to be used for PBR tests. This overrides explorer.ipv6_test.

See also: peer.X.ipv4_pbr_check.

Possible values: valid IPv6 address

4.14.44 peer.X.limit_load

Defines the load limit for current provider. IRP will improve routes to this provider as long as its load is less than the specified value in megabits per second. SNMP must be properly configured in order for this feature to work properly. If this limit is configured, but it is impossible to retrieve interface data, no new improvements will take place.

The value of peer.X.limit_load(if set to positive value) must be greater or equal to value of peer.X.95th.

Possible values: `-1, 1-1000000` -1 means unlimited

Default value: `-1`

Recommended value: it is recommended to set it to not more than ~60-80% of the physical interface rate

See also: peer.X.snmp.interfaces

4.14.45 peer.X.mon.ipv4.bgp_peer

Defines the current provider's IPv4 address that must be monitored by Bgpd, usually equal to peer.X.ipv4.next_hop

Possible values: valid IPv4 address

See also: peer.X.mon.snmp

4.14.46 peer.X.mon.ipv4.external.state

Enables or disables external monitors for IPv4.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.14.47 peer.X.mon.ipv4.internal.mode

Specifies router supported BGP MIB to monitor this provider IPv4 BGP sessions.

- 0 - Generic (BGP4-MIB)
- 1 - Cisco (CISCO-BGP4-MIB)
- 2 - Juniper (BGP4-V2-MIB-JUNIPER)
- 3 - Brocade (draft-ietf-idr-bgp4-mibv2-12)
- 4 - Huawei (HUAWEI-BGP-VPN-MIB)
- 5 - Arista (ARISTA-BGP4V2-MIB)
- 6 - Dell (DELLEMC-OS10-BGP4V2-MIB)

Possible values: `0, 1, 2, 3, 4, 5, 6`

Default value: `0`

4.14.48 peer.X.mon.ipv4.internal.state

Enables or disables internal monitors for IPv4.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.14.49 peer.X.mon.ipv6.bgp_peer

Defines the current provider's IPv6 address that must be monitored by Bgpd, usually equal to peer.X.ipv6.next_hop

Possible values: valid IPv6 address

See also: peer.X.mon.snmp

4.14.50 peer.X.mon.ipv6.external.state

Enables or disables external monitors for IPv6.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.14.51 peer.X.mon.ipv6.internal.mode

Specifies router supported BGP MIB to monitor this provider IPv6 BGP sessions.

- 1 - Cisco (CISCO-BGP4-MIB)
- 2 - Juniper (BGP4-V2-MIB-JUNIPER)
- 3 - Brocade (draft-ietf-idr-bgp4-mibv2-12)
- 4 - Huawei (HUAWEI-BGP-VPN-MIB)
- 5 - Arista (ARISTA-BGP4V2-MIB)
- 6 - Dell (DELLEMC-OS10-BGP4V2-MIB)

Possible values: `1, 2, 3, 4, 5, 6`

4.14.52 peer.X.mon.ipv6.internal.state

Enables or disables internal monitors for IPv6.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.14.53 peer.X.mon.snmp

Identifier of SNMP host to use for monitoring this provider. Refer to SNMP Host, snmp.X.name.

Possible values: `1-1000`

4.14.54 peer.X.mon.vpn_instance_id

The Provider routing instance index.
The parameter is useful when a provider belongs to a non-default routing instance, and the Internal Monitor should be addressed to monitor a BGP session inside that specific routing instance.

⚠ Applicable only to Arista, Dell and Huawei modes of Internal Monitor.
Default value depends on the mode: Huawei routers have the main routing instance ID equal to zero while Arista routers have this ID equal to 1.

Refer to peer.X.mon.ipv4.internal.mode, peer.X.mon.ipv6.internal.mode, Arista BGPv2 MIB, Huawei BGP VPN MIB.

Possible values: `0-4294967295`

4.14.55 peer.X.pbr_check

Enables/disables PBR check for a specific provider.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

Recommended value: `1`

4.14.56 peer.X.precedence

Precedence value of “0” has a special meaning. IRP excludes providers from grouping if a provider’s precedence value is set to 0.

Defines this provider’s precedence, used for provider grouping and defining commit priorities.
See core.commit_control and Commit Control for Commit Control functionality.

All providers belonging to a single group must have the same value for the following parameters:

- peer.X.95th.bill_day
- peer.X.95th.mode
- peer.X.95th.centile
- peer.X.cc_disable
- peer.X.cost
- peer.X.global_group
- peer.X.group_loadbalance
- peer.X.improve_in_group

If two different providers have the same peer.X.precedence value (except 0), then these providers are considered to be a group, and traffic is balanced between them (refer to peer.X.group_loadbalance).

The provider or a group with the lowest precedence value are also used by Commit Control as last resort destination (if all the providers are exceeding their 95th, then traffic is rerouted to the upstream with the smallest precedence - usually this upstream has either the highest throughput, or the lowest cost).

For example:

```
peer.1.precedence = 20
peer.2.precedence = 10
peer.3.precedence = 30
```

If the peer.X.95th is exceeded for all configured providers, then the excessive traffic from providers 1 and 3 will be rerouted to the 2nd provider.

If provider groups are used:

```
peer.1.precedence = 50
peer.2.precedence = 40
peer.3.precedence = 40
peer.4.precedence = 40
peer.5.precedence = 30
```

Traffic between providers 2, 3 and 4 is evenly distributed. If all providers are already overloaded, then the excessive traffic will be rerouted to the 5th provider (since it has the lowest precedence value).

Possible values: `0-100`

Default value: `0`

4.14.57 peer.X.rd

Assigns a routing domain identifier to the provider. Providers in the same routing domains should be assigned the same identifier.

Providers with identifier 1 (one) are assumed to be in the same routing domain that hosts IRP instance.

Refer global.rd_rtt, peer.X.flow_agents, bgpd.rd_local_mark.

Possible values: `1-100`

Default value: `1`

4.14.58 peer.X.routes_config

Defines whether Internet Exchanges auto configuration is enabled or not. In case it is enabled, IRP Bgpd gathers Peering Partners (next-hops) with their announced routes and stores the data in IRP database. Otherwise, manual configuration is required.

Possible values: `0 (Manual configuration), 1 (Autoconfiguration from BGP)`

Default value: `1`

4.14.59 peer.X.shortname

⚠ Mandatory

Defines the provider's short, abbreviated name (3-20 characters). This parameter's value is used for reports and graphs.

Possible values: text

4.14.60 peer.X.shutdown

Defines whether provider is Active (0), Suspended (1) or Shutdown (2).

After the provider suspend action, all the improvements will be stored for short period of time (1h). In case of short maintenance windows, use provider suspend.

After the provider shutdown action, all the improvements will be removed. In case of long maintenance windows, use provider shutdown.

After the provider reactivation (after a previous suspend), all the improvements will be sent to retry probing.

Possible values: `0, 1, 2`

Default value: `0`

4.14.61 peer.X.snmp.enhanced

Enables or disables heuristically enhanced SNMP collection for provider.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.14.62 peer.X.snmp.interfaces

Defines a collection of interfaces used for current provider.

There have been four formats of individual values where the first number represents identifier of an SNMP host configured under 4.19, the delimiter depicts the type of matching algorithm, and the value after the delimiter is used to access the required interface stats:

id-number Match interface by ifIndex

id=name Match interface by ifName

id:description Match interface by ifDescr

id|alias Match interface by ifAlias

Possible values: collection of SNMP Interfaces

4.14.63 peer.X.type

Defines the type of a provider. The following types are available:

- 0 - Transit provider
- 1 - Partial routes provider
- 2 - Exchanges provider

Possible values: `0, 1, 2`

Default value: `0`

4.15 Inbound rule

4.15.1 inbound.rule.X.bgp_peer

Defines the router(s) where IRP announces improvements of this inbound prefix.

Possible values: `a router identifier(s) separated by space`

4.15.2 inbound.rule.X.communities

The list of additional BGP communities that will be set for the Inbound Prefix.

4.15.3 inbound.rule.X.enabled

Enables or disables Inbound Optimization for this prefix.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.15.4 inbound.rule.X.full_control

Specifies individual inbound prefix control level by IRP. Default behavior can be set at system level under bgpd.full_control. The settings specify whether default behavior is inherited from the system level or set individually for this prefix to announce either Improvements only or fully announce the prefix to allowed providers.

Possible values: `-1 (Use default), 0 (Improvements), 1 (If improved), 2 (All)`

Default value: `-1`

4.15.5 inbound.rule.X.next_hop

Defines a next_hop specific for the inbound rule. When the next_hop is unset, then a value from bgpd.peer.X.inbound.ipv4.next_hop/bgpd.peer.X.inbound.ipv6.next_hop is taken.
Next-hop should point to a router that routes/terminates traffic. Otherwise null route should be configured for the Next-Hop.

> next_hop value should be of the same IP address family as the inbound prefix it is assigned to. Refer inbound.rule.X.prefix.

Possible values: `IP address`

4.15.6 inbound.rule.X.prefix

Defines an inbound prefix. Inbound prefixes should belong to your network.

> Inbound Optimization of transit networks will be supported in future releases.

> Each inbound prefix logically extends the list of ournets when it is processed by IRP Collector. Refer collector.ournets.

Possible values: `prefix in CIDR format`

4.15.7 inbound.rule.X.providers

Defines the list of providers where this inbound prefix can be advertised. In case this parameter value is empty then it is treated as prefix will be advertised to ALL providers.

Possible values: `providerIDs collection`

4.16 Routing Policy

> IRP used a legacy format for policies.conf where a single policy parameters immediately followed the prefix/ASN that defined them. IRP updated policies.conf format and assigned each policy a unique identifier of the form "policy.X." used to prefix each attribute. Notice the format changes in the samples below.

Legacy routing policy example:

```
asn=65530
vip=0
policy=deny
providers=1 3 5
enabled=1
notes=AS65530 deny via Level3, Cogent and nLayer
```

Normalized routing policy example:

```
policy.1.asn=65530
policy.1.vip=0
policy.1.policy=deny
policy.1.providers=1 3 5
policy.1.enabled=1
policy.1.notes=AS65530 deny via Level3, Cogent and nLayer
```

4.16.1 policy.X.asn

Defines the Autonomous System Number which is used to match prefixes from the BGP routing table, originated from the specified ASN.

> Only one matching parameter should be specified per policy

Possible values: `1-4294967295`

4.16.2 policy.X.cascade

Defines the ASN policies that cascade to downstream AS from designated AS. The parameter applies to ASN policies (and not to prefixes).

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.16.3 policy.X.cc_overload_by_latency

This parameter grants latency optimizations the ability to override Commit Control decisions.

When enabled, it instructs IRP to prioritize latency reduction over strict adherence to predefined bandwidth controls.

The dynamic adjustment enhances performance, ensuring faster data transmission even if it means deviating from established bandwidth distribution policies among providers or provider groups.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.16.4 policy.X.community

A community to mark improvements belonging to a policy.

Possible values: `X:Y`

4.16.5 policy.X.country

Defines the two-character country code which is used to match prefixes from the internal geodata database.

> Only one matching parameter should be specified per policy

Possible values: `Two-character ISO country Code`

4.16.6 policy.X.enabled

Defines whether the policy is enabled or not

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.16.7 policy.X.forcelocal

If the parameter is enabled then global improvements aren't performed.
Refer: policy.X.policy

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.16.8 policy.X.notes

Defines a description of the routing policy

Possible values: `text`

4.16.9 policy.X.policy

Defines the type of policy.

allow IRP ensures that traffic flows only via providers listed in the policy.X.providers parameter

deny IRP ensures that traffic does not flow via providers listed in policy.X.providers parameter

static IRP selects prefixes matched by a corresponding parameter from the BGP routing table and makes static improvements via provider(s) specified in the policy.X.providers parameter.

static_exact This option can be enabled when the policy.X.prefix parameter gets used.

Possible values: `allow, deny, static, static_exact`

Default value: `allow`

4.16.10 policy.X.prefix

Defines the aggregate prefix which is used to match sub-prefixes from the BGP routing table.
If the policy.X.policy parameter is set to "static_exact" then the prefix is used "as-is" to make a route via specified provider(s).
If the policy.X.policy parameter is set to "static" or "static_exact" and the policy.X.forcelocal parameter is enabled, then multiple providers (one per routing domain) can be specified in the policy.X.providers parameter.
Refer: policy.X.forcelocal, policy.X.policy

> Only one matching parameter should be specified per policy

Possible values: `Valid IPv4 or IPv6 prefix`

4.16.11 policy.X.priority

Defines what policy to prioritize in case of overlapping prefixes as might be the case of a large aggregate prefix policy extending over an ASN policy. The same specific prefixes might be covered by both and priority sets which one to actually enforce in favor of the others. Policies with higher priority are prioritized.

Possible values: `0-1000`

Default value: `0`

4.16.12 policy.X.providers

Defines the list of providers (IDs) affected by the policy. If the 'providers' property is not specified, then all the providers will be included in the policy by default.

Possible values: `1-unlimited`

4.16.13 policy.X.vip

Defines whether VIP probing is enabled for the specified policy.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.17 Exchange

4.17.1 provider.X.rule.Y.bgp_peer

Defines the BGP Router-ID used for BGP session state monitoring, which is performed by the Internal BGP Monitor (BGP Monitoring).

Depending on whether a Route Server or Peering sessions are used, the parameter value should be set to Route Server Router-ID or Peering Partner Router-ID accordingly.

Possible values: `IPv4 address`

4.17.2 provider.X.rule.Y.enabled

Defines whether the rule is enabled or not.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.17.3 provider.X.rule.Y.next_hop

Defines the Peering Partner's IPv4/IPv6 next-hop address.

Possible values: `IPv4/IPv6 address`

4.17.4 provider.X.rule.Y.next_hop_as

Defines the peering partner's autonomous system number. This parameter must be set in order for the 3rd algorithm of Bgpd AS-PATH to work properly (refer to bgpd.as_path).

If this parameter is set, then it's value will be used as part of AS-PATH for outgoing improvements if the 3rd algorithm is enabled in bgpd.as_path.

In case of Internet Exchanges, the AS-Path begins with peering partner's AS number, instead of AS number of route server.

The first AS number will be stripped from AS-Path when advertising improvement towards Exchange, in case the first AS number is equal to the value set in this parameter.

See also: peer.X.ipv4.next_hop_as, peer.X.ipv6.next_hop_as

Possible values: `0-4294967295`

Default value: `0`

4.17.5 provider.X.rule.Y.pbr_check

Enables/disables PBR check for a configured peering parther.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `1`

4.17.6 provider.X.rule.Y.probing_dscp

Defines the DSCP (Differentiated services code point) used with provider.X.rule.Y.probing_ip for a specific Peering Partner.

Possible values: `0-63`

4.17.7 provider.X.rule.Y.probing_ip

Defines the probing IPv4/IPv6 address used for a specific Peering Partner.

Possible values: `IPv4/IPv6 address`

4.17.8 provider.X.rule.Y.shortname

Defines a short name for a configured Peering Partner.

Possible values: `text`

4.17.9 provider.X.rule.Y.transit

The option could be turned on if a Peering Partner provides full Internet access.
IRP will check for route availability, assuming a route is always available.

Possible values: `0 (Disabled), 1 (Enabled)`

Default value: `0`

4.18 Global Group

4.18.1 globalgroup.X.members

Defines a list of IP addresses or host names of all IRP instances (including Failover slave nodes) to form the Global Group.

⚠ Value of the parameter must be the same on all instances of the Global Group.

4.18.2 globalgroup.X.outbound.95th

Amount of bandwidth provided by this IRP instance to the Global Group.
If the value -1 is set, then IRP automatically uses total amount of outbound bandwidth set in the peer.X.95th parameter for all the providers in the Global Group.

Possible values: `-1, 1-1000000`

4.18.3 globalgroup.X.outbound.95th.reserve

Amount of bandwidth reserved to operation of this IRP instance.

Possible values: `0-1000000` 0-(not used)

Default value: `0`

4.18.4 globalgroup.X.outbound.rate_high

Defines Global Group's high load rate limit (%).

⚠ Value of the parameter must be the same on all instances of the Global Group.

See also: core.commit_control.rate.high

Possible values: `50-99`, but greater than or equal to globalgroup.X.outbound.rate_low

Default value: `95`

4.18.5 globalgroup.X.outbound.rate_low

Defines Global Group's low load rate limit (%).

⚠ Value of the parameter must be the same on all instances of the Global Group.

See also: core.commit_control.rate.low

Possible values: `30-99`, but lower than or equal to globalgroup.X.outbound.rate_high

Default value: `85`

4.18.6 globalgroup.X.shortname

Defines short name of the Global Group.

⚠ Value of the parameter must be the same on all instances of the Global Group.

4.19 SNMP Host

4.19.1 snmp.X.auth_password

User's password used to authenticate SNMP communications. Applicable only when SNMP v3 with authentication is used. Refer to snmp.X.version, snmp.X.seclevel, snmp.X.auth_username.

Possible values: `text`

4.19.2 snmp.X.auth_protocol

SNMP authentication protocol. Applicable only when SNMP v3 with authentication is used. Refer to snmp.X.version, snmp.X.seclevel.

- 0 - MD5
- 1 - SHA

Possible values: `0, 1`

4.19.3 snmp.X.auth_username

SNMP user name. Applicable only when SNMP v3 with authentication is used. Refer to snmp.X.version, snmp.X.seclevel.

Possible values: `text`

4.19.4 snmp.X.community

⚠ Mandatory parameter

Defines the SNMP community.

Possible values: textual community name

4.19.5 snmp.X.context

Defines the SNMPv3 context name.

Possible values: textual context name

4.19.6 snmp.X.ip

Defines IPv4/IPv6 address of the SNMP host.

⚠ Hostnames are not supported.

Possible values: valid IPv4 or IPv6 address

4.19.7 snmp.X.name

Sets a shortname for SNMP host.

Possible values: text

4.19.8 snmp.X.oids_per_pdu

Defines the maximum number of OIDs that can be requested in a single PDU. Refer bgpd.snmp.packets_interval.

Possible values: `1-300`

Default value: `10`

4.19.9 snmp.X.port

Defines the UDP port to send SNMP requests.

Possible values: `1-65535`

Default value: `161`

4.19.10 snmp.X.priv_password

Password used to encrypt SNMP communications. Applicable only when SNMP v3 with privacy is used. Refer to snmp.X.version, snmp.X.seclevel.

Possible values: `text`

4.19.11 snmp.X.priv_protocol

SNMP encryption protocol. Applicable only when SNMP v3 with privacy is used. Refer to snmp.X.version, snmp.X.seclevel.

> DES isn't supported in RedHat Enterprise Linux 9

- 0 - DES
- 1 - AES

Possible values: `0, 1`

4.19.12 snmp.X.seclevel

SNMP security services. IRP supports either No security, Authentication only or both Authentication and Privacy. Applicable only when SNMP v3 is used (snmp.X.version). Depending on security services used further parameters should be configured, for example: snmp.X.auth_username, snmp.X.auth_password, snmp.X.auth_protocol, snmp.X.priv_password, snmp.X.priv_protocol.

- 0 - None - neither Authentication nor Privacy is used
- 1 - Authentication only
- 2 - Authentication and Privacy

Possible values: `0, 1, 2`

Default value: `0`

4.19.13 snmp.X.timeout

Timeout period before retrying an SNMP request (s).

Possible values: `1-25`

Default value: `1`

4.19.14 snmp.X.version

SNMP version used by host.

- 2 - SNMP v2c
- 3 - SNMP v3

Possible values: `2, 3`

Default value: `2`

4.20 Routing domain

4.20.1 rd.X.community.local

Specifies a BGP community which will be appended to BGP community attribute of an improvement in the corresponding routing domain.

Parameter represents a valid value for BGP community attribute of the form X:Y. Value in rd.X.community.local is APPENDED to communities attribute.

⚠ The value must be unique across all configured BGP communities.

Refer global.rd_rtt, peer.X.rd, peer.X.flow_agents, bgpd.rd_local_mark.

Possible values: `X:Y`

4.20.2 rd.X.community_worsening

RD performance worsening community.

The routing domain designated community value is added to the announced global outbound improvement if such improvement resulted in a degraded performance per the allowed worsening thresholds for a specific domain.

Possible values: `valid BGP community`

4.20.3 rd.X.shortname

A shortname assigned to each routing domain for human readability.

Possible values: `text`

Chapter 5

Appendixes

5.1 Frontend labels for configuration parameters

API allowed addressesapid.allowed_ips
APId listen IPv4/IPv6 addressapid.listen.master_ip
APId listen IPv4/IPv6 address (slave)apid.listen.slave_ip
AS Path rules for IX ASNpeer.X.aspath_for_ix
AS-PATH restore prioritybgpd.as_path
Account IDpushd.sms.account_sid
Adaptive packets countexplorer.probing.sendpkts.adaptive_max
Additional communitiesinbound.rule.X.communities
Allow Latency to override CCpolicy.X.cc_overload_by_latency
Allowed IP addressesglobal.frontend_acl_ips
Allowed latency worsening (ms)core.cost.worst_ms
Allowed loss worsening (%)core.commit_control.worst_loss
Alternative PBR test IPv4 addresspeer.X.ipv4_pbr_check
Analyzed prefixescollector.ournets
Announced blackholing localpref valuebgpd.peer.X.blackholing.localpref
Announced inbound localpref valuebgpd.peer.X.inbound.master_localpref, bgpd.peer.X.inbound.slave_localpref
Autonomous Systembgpd.peer.X.as
Autonomous system numberpolicy.X.asn
Avatar emojipushd.webhook.avatar_emoji
Avatar icon URLpushd.webhook.avatar_url
BGP MIB (IPv4)peer.X.mon.ipv4.internal.mode
BGP MIB (IPv6)peer.X.mon.ipv6.internal.mode
BGP Router IDbgpd.peer.X.master_router_id, bgpd.peer.X.slave_router_id
BGP communitypolicy.X.community
BGP modeglobal.nonintrusive_bgp
BGP redirect communitiesirpdetectd.bgp.redirect.communities
BGP redirect routersirpdetectd.bgp.redirect.bgp_peers
BGP session monitoring IPv4 addresspeer.X.mon.ipv4.bgp_peer
BGP session monitoring IPv6 addresspeer.X.mon.ipv6.bgp_peer
BGP session passwordbgpd.peer.X.master_password, bgpd.peer.X.slave_password
BW reserved for local instance operationglobalgroup.X.outbound.95th.reserve
Base community for inbound improvementspeer.X.inbound.community_base
Bgpd monitoring guard time (sec)bgpd.mon.guardtime

Bgpd monitoring holdtime (sec)bgpd.mon.holdtime
Bgpd monitoring keepalive (sec)bgpd.mon.keepalive
Bgpd monitoring long holdtime (sec)bgpd.mon.longholdtime
Blackhole threshold kppsirpdetectd.blackhole.threshold.kpps
Blackhole threshold mbpsirpdetectd.blackhole.threshold.mbps
Blackholing IPv4 next hopbgpd.peer.X.blackholing.ipv4.next_hop
Blackholing IPv6 next_hopbgpd.peer.X.blackholing.ipv6.next_hop
Blackholing Routerspeer.X.blackholing.bgp_peer
Blackholing communitypeer.X.blackholing.community
Borrow AS-PATH if can't restorebgpd.as_path_borrowing
Bot namepushd.webhook.botname
CC allowed forcore.commit_control.loss_override
CC probing TTL (sec)core.commit_control.probe_ttl
CC probing queue slotscore.commit_control.probing_queue_size
CC provider precedencepeer.X.precedence
Cascade policypolicy.X.cascade
Cascading policy max ASbgpd.policy.cascade.amount
Centile valuepeer.X.95th.centile
Circuit issues detectionpeer.X.circuit.control
Commit Controlcore.commit_control
Commit Control for inboundcore.commit_control.inbound.enabled
Commit Control for providerpeer.X.cc_disable
Community mark for RDrd.X.community.local
Community marker for local improvementsbgpd.rd_local_mark
Confirmation reprobing intervalirpinperfd.probing.confirmation_interval
Controlinbound.rule.X.full_control
Countrypolicy.X.country
DDoS Modeirpdetectd.mode
Default BGP reactionirpdetectd.bgp.reaction
Default FlowSpec reactionirpdetectd.flowspec.reaction
Delete irrelevant CC improvementscore.commit_control.del_irrelevant
Delta loss to restorecore.circuit.recover_loss_diff
Delta loss to shutdowncore.circuit.high_loss_diff
Delta loss to warncore.circuit.warn_loss_diff
Diag hop uniqueness enforcementexplorer.trace.diag_hop_unique
Email from addresspushd.email.from
Enable policypolicy.X.enabled
Explorer algorithmsexplorer.algorithms
Explorer worker threadsexplorer.maxthreads
Exploring IPv4 queue slotscore.eventqueuelimit
Exploring IPv6 queue slotscore.eventqueuelimit_ipv6
Failback timer (s)global.failover_timer_failback
Failed probe lifetime (sec)core.probes.ttl.failed
Failoverglobal.failover
Failover master identity fileglobal.failover_identity_file
Failover timer (s)global.failover_timer_fail
Flapping protectionbgpd.mon.internal.flap_guardtime

Mitigation prefix size (IPv6)irpdetectd.ipv6.prefix_size
Model shelf life (s)irpinperfd.model.shelf_life
Model stability check interval (s)irpinperfd.model.stability_interval
NO_EXPORT/NO_ADVERTISE communitybgpd.no_export
NetFlow UDP portcollector.flow.listen.nf
Outage confirmation timeout (sec)core.problem.outage_timeout
Outage detectioncore.outage_detection
Outage prefix confirmation ratecore.outage_detection.limit_pct
Outage round trip rate (%)core.problem.rtt.diff_pct
Outbound high load limit(%)core.commit_control.rate.high
Outbound low load limit(%)core.commit_control.rate.low
Outbound optimizationglobal.outbound
Outbound traffic matchingcollector.flow.all_outbound
Overload by (%)trap.core_cc_overload.limit_pct
Overload by (Mbps)trap.core_cc_overload.limit_mbps
Overusage interval (sec)core.overusage.check_interval
Overusage policiesglobal.bw_overusage
Overusage rule retention (sec)core.overusage.hold_timer
Overusage threshold multipliercore.overusage.out.threshold.trigger
Overusage throttle multipliercore.overusage.out.threshold.throttle
PBR check for providerpeer.X.pbr_check
Packet size from IP headercollector.span.size_from_ip_header
Peer AS number overrideprovider.X.rule.Y.next_hop_as
Peering Partner PBR checkprovider.X.rule.Y.pbr_check
Peering Partner nameprovider.X.rule.Y.shortname
Peering Partner next-hopprovider.X.rule.Y.next_hop
Peering Partner statusprovider.X.rule.Y.enabled
Perf/Cost improvements within grouppeer.X.improve_in_group
Performance for outboundglobal.outbound.performance
Plivo API uripushd.sms.uri.plivo
Policy notespolicy.X.notes
Policy prioritypolicy.X.priority
Policy typepolicy.X.policy
Prefixinbound.rule.X.prefix
Prefix BW average time (hours)core.overusage.out.average.period
Prefix aggregationglobal.aggregate
Prefix announcement rate (%)trap.bgpd_announced_rate_low.limit_pct
Prefix filter for excessive loss/latencytrap.core_excessive.prefixes
Prefix latency limit (ms)trap.core_excessive_latency.limit
Prefix loss limit (%)trap.core_excessive_loss.limit
Prefix relevant BW (Mbps)core.overusage.out.average.relevant_min
Prefix statusinbound.rule.X.enabled
Prepend inbound prefixescore.circuit.inbound
Prepend transit prefixescore.circuit.transit
Probe shelf life (s)irpinperfd.probing.shelf_life
Probing DSCPprovider.X.rule.Y.probing_dscp
Probing IP addressprovider.X.rule.Y.probing_ip

Probing IPv4 addresspeer.X.ipv4.master_probing, peer.X.ipv4.slave_probing
Probing IPv6 addresspeer.X.ipv6.master_probing
Probing algorithmexplorer.probe.algorithm
Probing failure marginirpinperfd.probing.failure_margin
Probing interface(s)global.master_probing_interface
Probing timeoutirpinperfd.probing.timeout
Protected addressesirpdetectd.protected_addresses
Provider 95th Calculation Modepeer.X.95th.mode
Provider 95th percentilepeer.X.95th
Provider ASN for IPv4peer.X.ipv4.next_hop_as
Provider ASN for IPv6peer.X.ipv6.next_hop_as
Provider Shutdownpeer.X.shutdown
Provider cost per Mbps (USD)peer.X.cost
Provider descriptionpeer.X.description
Provider inbound overload by (%)trap.core_cc_provider_overload.inbound_limit_pct
Provider overload by (%)trap.core_cc_provider_overload.limit_pct
Provider overload by (Mbps)trap.core_cc_provider_overload.limit_mbps
Provider overload inbound by (Mbps)trap.core_cc_provider_overload.inbound_limit_mbps
Provider probing IPv6 addresspeer.X.ipv6.slave_probing
Provider typepeer.X.type
Provider's 95th (inbound)peer.X.95th.in
Provider's bandwidth max deviation (%)core.commit_control.rate.group
Provider's billing daypeer.X.95th.bill_day
Provider's global group IDpeer.X.global_group
Provider's routing domainpeer.X.rd
Provider/link namepeer.X.shortname
Providersinbound.rule.X.providers
Public IPv4 address for PBR testsexplorer.ipv4_test
Public IPv6 address for PBR testsexplorer.ipv6_test, peer.X.ipv6_pbr_check
RD performance worsening communityrd.X.community_worsening
RD shortnamerd.X.shortname
RTT allowed worsening (ms)irpinperfd.rtt_allowed_worsening
RTT between RDsglobal.rd_rtt
Re-probe on new path changebgpd.retry_probing.new.bmp_path_change
Re-probe on old path changebgpd.retry_probing.old.bmp_path_change
React on Irpflowd statscore.commit_control.react_on_collector
Relevant RTT difference (%)core.performance.rtt.diff_pct
Relevant RTT difference (ms)core.performance.rtt.diff_ms
Relevant RTT: IX vs transit (%)core.performance.rtt.ix_diff_pct
Relevant RTT: IX vs transit (ms)core.performance.rtt.ix_diff_ms
Relevant loss (%)core.performance.loss_pct
Remove if exact prefix absent in BMPbgpd.improvements.remove.bmp_exact_aggregate
Remove next hop updatebgpd.improvements.remove.next_hop_eq
Remove on aggregate withdrawalbgpd.improvements.remove.withdrawn
Reprobing intervalirpinperfd.probing.interval
Restore after (sec)core.circuit.recover_hold_time
Restore interval (min)core.circuit.recover_monitored_intervals

Speaking IP responses for candidatesexplorer.scanning.replypkts.min
Speaking IPs RTT dispersion (ms)explorer.scanning.rtt.dispersion_ms
Speaking IPs to scan on lossexplorer.scanning.confirm_ips
Spike preceeding interval (seconds)trap.core_cc_improvements_spike.period_sec
Spike size (%)trap.core_cc_improvements_spike.diff_pct
Split outbound announcementsbgpd.updates.split
Standard reprobing period (sec)core.improvements.ttl.retry_probe
Strip non-IRP communitiesbgpd.improvements.strip_non_irp_communities
Synchronous Inbound Provider Groupscore.commit_control.inbound.sync_groups
TCP packets interval (ms)explorer.interval.tcp_syn
TCP port collection modecollector.flow.tcp_ports.mode
TCP port number capcollector.flow.tcp_ports.limit
TCP port range endcollector.flow.tcp_ports.max
TCP port range startcollector.flow.tcp_ports.min
TCP ports listcollector.flow.tcp_ports.list
The routing instance indexpeer.X.mon.vpn_instance_id
Time keepirpdetectd.time.keep
Time monitorirpdetectd.time.monitor
Timeout interval (s)snmp.X.timeout
Top volume prefixes per cyclecollector.export.volume.high.top_n
Top volume prefixes per inperf ruleirpinperfd.model.topn_per_rule
Top-N relevant volume prefixescore.outage_detection.volume_top_n
Trace all providersexplorer.trace.all
Traceroute algorithms orderexplorer.trace.algorithms
Traceroute max hopsexplorer.traceroute.ttl.max
Traceroute min hopsexplorer.traceroute.ttl.min
Traceroute packets per hopexplorer.traceroute.sendpkts
Traceroute retry packetsexplorer.traceroute.retrypkts
Transit ASNsbgpd.prefixlist.asn
Transit Improvement TTL max (sec)core.improvements.inbound_transit.ttl.max
Transit Improvement TTL min (sec)core.improvements.inbound_transit.ttl.min
Transit Improvements Maxcore.improvements.inbound_transit.max
Transit SNMP hostsbgpd.peer.X.transit.snmp
Transit communitiesbgpd.transit.communities
Transit matching at egresscollector.flow.process_transit_in_outbound
Transit peering partner markprovider.X.rule.Y.transit
Transit prefixesbgpd.prefixlist.prefixes
Transit top N prefixescollector.flow.export.inbound_transit.topn
Transit traffic distribution historycollector.export.archive_transit
Transiting trafficbgpd.peer.X.transit.status, global.inbound_transit
Twilio API uripushd.sms.uri.twilio
Use BGP peer for flowspec PBRpeer.X.flowspec.pbr.use_bgp_peer
Use BMP datapeer.X.bmp
Use BMP to search for routespeer.X.bmp.check_routes
Use communities to monitor masterglobal.failover.use_communities
VIP probingpolicy.X.vip
VIP reprobing period (sec)core.vip.interval.probe

Webhook URLpushd.webhook.url

Whitelistirpdetectd.whitelist

Withdraw improvements on warncore.circuit.withdraw_on_warn

min_delay statuscollector.span.min_delay

sFlow UDP portcollector.flow.listen.sf

5.2 Configuration parameter index

glo

com

bgp

bgp1

col

cor

exp

adm

ups

rou

exc

tra

api

inb

snmp

rou1

bmp

glo1

inb1

tms