Technical Guide

Software Defined WAN (SD-WAN)

Feature Overview and Configuration Guide

Introduction

This guide describes the Allied Telesis™ Software Defined Wide Area Network (SD-WAN) solution and how to configure it. SD-WAN is a technology which increases performance and/or control of application traffic over redundant WAN interfaces and VPN links. The Allied Telesis SD-WAN solution is able to make routing decisions based on the quality of virtual private network (VPN) links. To do this, it uses link probing to determine path quality and perform application-aware routing to dynamically redirect performance sensitive application traffic (for example voice or video) via redundant links that meet application performance requirements.

List of Terms:

SD-WAN
Software Defined Wide Area Network

PBR
Policy-Based Routing

DPI
Deep Packet Inspection

VPN
Virtual Private Network

VTI
Virtual Tunnel Interface

FQDN
Fully Qualified Domain Name

UTM
Unified Threat Management

AlliedWare Plus™ OPERATING SYSTEM

Contents

Products and software version that apply to this guide

This guide applies to AlliedWare Plus™ products that support SD-WAN, running version **5.4.8-1.0** or later.

To see whether your product supports SD-WAN, see the following documents:

- The product's Datasheet
- The product's Command Reference

These documents are available from the above links on our website at alliedtelesis.com.

What is SD-WAN?

SD-WAN is a series of features working together to provide various solutions that meet modern business intent and reduce costs.

Modern applications like Voice over IP (VoIP) calling, videoconferencing, streaming media, and virtual applications and desktops need low latency. Bandwidth requirements for applications are also increasing. Expanding WAN capability can be expensive, and dealing with network management and troubleshooting can be difficult.

SD-WAN lets service providers and enterprises use existing physical customer-premises equipment (CPE) and utilize redundant low-cost WAN and VPN connections. This lets you create fully managed multi-site networks, integrating links and optimizing application flows to the Internet and across the enterprise VPN infrastructure.

The four pillars of SD-WAN

There are four fundamental ideas underlying SD-WAN.

Transport independence

SD-WAN can utilize different connection types, such as wireless, commodity broadband, multi-protocol label switching (MPLS), and so on. A VPN can then be overlaid on each WAN connection to provide security and privacy. Application traffic over these VPN links should be automatically distributed and load-balanced, with no manual intervention required. This avoids the need for expensive dedicated links.

Application optimization

SD-WAN ensures that crucial applications can have their bandwidth and other metrics met. This requires a rich application-based and statistical dataset to allow you to manage the enterprise network. Once you can see how the applications are currently performing in the network, you can architect and modify the network and application forwarding paths to meet those needs.

Intelligent path control

This is the core of SD-WAN. The router allows you to dynamically determine the best path for each application's traffic. For example, voice traffic will prefer the link that has the lowest latency and jitter at all times, regardless of the underlying connection. Other applications such as YouTube, Facebook, or file-transfers can be prioritized to use cheaper links. This is achieved using probe metrics.

Secure connectivity

Everything needs to be encrypted when transported over the public network between private sites. Using VPN links provides this security.

What are the benefits of SD-WAN?

SD-WAN offers a number of advantages over traditional WAN solutions.

- Build higher-performance WANs using lower-cost and commercially available Internet access. This lets you partially or entirely replace more expensive private WAN connection technologies such as MPLS.
- Reduce costs and mitigate risks. You can select any type of WAN connectivity to help lower costs without compromising security. Traffic can then be load-balanced across these tunnels to make optimal use of available bandwidth.
- Dynamic path selection allows administrators to set performance thresholds for different applications. This lets you ensure that critical applications and data transfers always use the best path based on the loss, latency, and jitter of the available VPN tunnels.
- Support for centralized monitoring and active device control. A centralized tool allows GUI-based provisioning of various SD-WAN services on SD-WAN devices operating within the enterprise. You can also monitor the health and traffic of your network.
  Vista Manager EX provides this functionality for Allied Telesis SD-WAN networks. For more information about Vista Manager EX, and using it to manage your SD-WAN network, refer to the Vista Manager EX Installation and User Guide.

Overview of SD-WAN Load Balancing

SD-WAN load balancing uses link probing and application-aware routing to optimize application traffic. It does this by dynamically redirecting performance sensitive traffic via redundant links that meet its performance requirements.

Certain types of traffic have very specific performance requirements to work correctly. High latency or jitter can severely degrade user experience for applications or devices which stream voice or video traffic. Often, cheap Internet links are able to meet the performance requirements for these applications. However, this is not guaranteed and performance can fluctuate dramatically. Because of this, businesses have been required to purchase expensive site-to-site WAN solutions such as MPLS, with bandwidth guarantees and committed information rates (CIR) high enough to meet their needs.

This SD-WAN load balancing solution minimizes or outright replaces the need for these expensive site-to-site WAN solutions, replacing them with one or more cheaper Internet WAN connections. Site-to-site connectivity is provided via VPNs traversing these Internet connections. It utilizes ICMP-based probes to determine the jitter, latency, packet-loss, and consecutive probe loss of redundant VPN tunnels between sites. When performance on a given tunnel no longer meets the performance requirements for a given application, it can be redirected onto a VPN which does.

Why use SD-WAN load balancing across VPNs?

The key requirement for SD-WAN load balancing is that the site (branch office) has redundant VPN connections to another location (head office). It's already possible to load balance traffic across two links using features such as equal cost multi-path (ECMP) and policy-based routing (PBR). However, there are significant limitations to these options.

ECMP hashes traffic flows across multiple links. This is effective, as long as the links are operational and the performance of each link isn't important. ECMP also assumes the hashing algorithm will result in a uniform spread of the traffic over multiple links. In practice, one link may become more heavily utilized than another, degrading overall performance.

PBR allows you to configure certain types of traffic to use a particular link. This gives you the ability to decide which link each type of application traffic should use. However, PBR alone can't take into account the current performance metrics of each link.

SD-WAN load balancing utilizes the ability of PBR to route traffic based on an application type. It adds to it the ability to dynamically change the routing path of an application based on the current performance metrics of each link. This means that high priority VoIP traffic can be configured so that the lowest latency link is always preferred. If the current link no longer meets the performance requirements, the traffic will be switched over to another link that does meet the requirements.

This means that SD-WAN allows an enterprise to buy several lower-cost WAN links for the branch office locations (for example, residential-grade Internet service) from multiple ISPs. This lets you achieve similar levels of service as provided by a single MPLS link with a dedicated bandwidth at a much cheaper price.

There are two key parts to SD-WAN load balancing. The first is the ability for each link to be monitored and real time performance metrics gathered. This is done by configuring a probe which will send ICMP packets across each link and measure the latency, jitter, packet-loss, and consecutive probe loss of each link. The second part is configuring per application routes along with their performance requirements. The router then continually monitors the performance metrics in real-time and chooses the best link to use for each application based on the current link metrics.

PBR routes traffic based on application type. To determine the application type, deep packet inspection (DPI) is used. This inspects the packets in a flow, and decides the flow’s application based on the contents of the packet.

Finding the best path

In networking, we rely on routing protocols to compute best path. Best path is typically computed using simplistic routing metrics like hop count, cost, and bandwidth. Given several paths to get to the destination, the best path is chosen based on the most desirable metric computed along those several paths.

SD-WAN brings a much more sophisticated metric to the computation of best path:

- Best path is computed and applied on a per-application basis. This is referred to as **application-aware routing**. The best path for a bulk traffic application might be different from the best path for a voice application, even if they have the same destination.
- Real-time link capabilities can be factored into the decision and traffic can be re-routed automatically if the quality of the link degrades below a specified threshold.
- Any link, no matter the available bandwidth compared to other links, can be used in an active/active manner.

How does SD-WAN differ from 'conventional' PBR?

SD-WAN makes heavy use of the existing concept of application-based PBR, so it's important to know what is different about SD-WAN. We refer to non-SD-WAN PBR rules as “conventional PBR” for clarity.

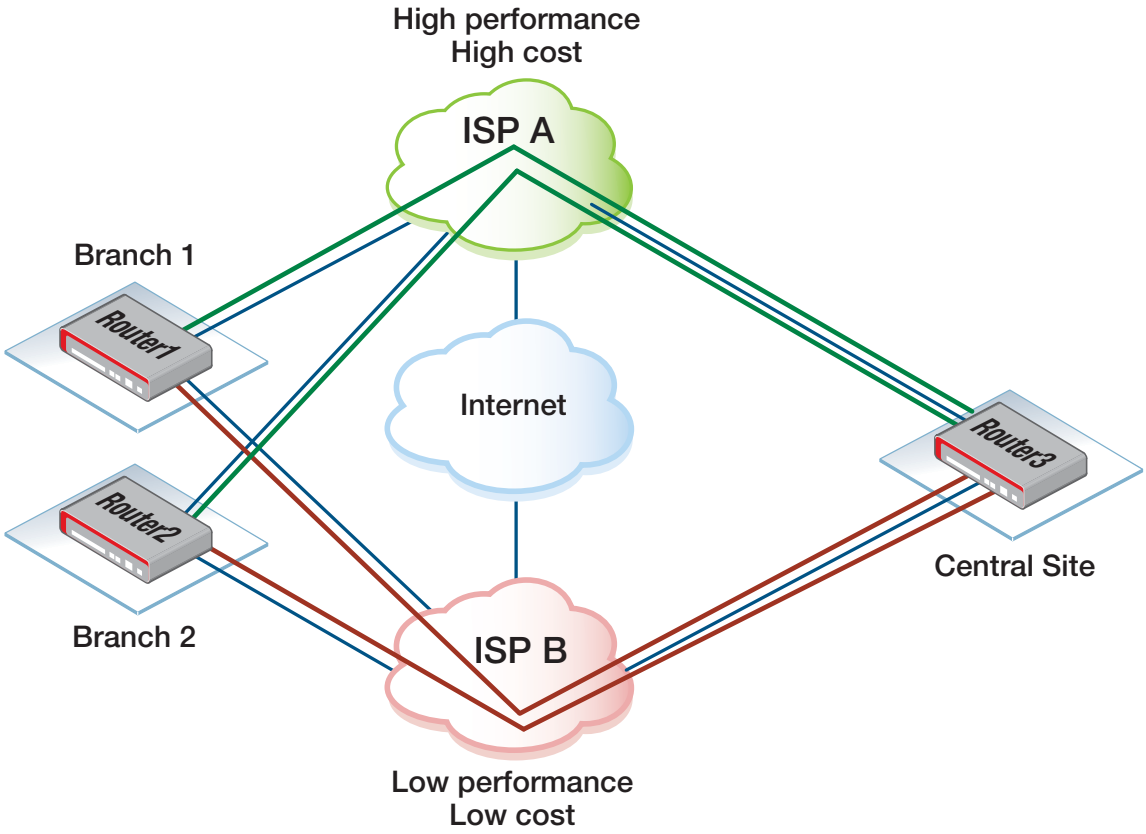
When you create a conventional PBR rule, you must supply a list of nexthops. The traffic that matches the rule will be routed to the first nexthop in the list that is link up. If you configure a list of three links, and link 1 goes down, link 2 will automatically be used instead. When link 1 comes back up it will again be re-used.

SD-WAN extends this concept beyond simply checking if a link is up or down. It can also use metrics about the health of the link to decide if the link is “good” or “bad”. This allows traffic to be re-directed from a “bad” link to a “good” link, even if both links are still up. The metrics that SD-WAN can use to judge the health of a link are jitter, latency, packet-loss, and consecutive probe loss. Each metric is examined separately, so that a link that is “bad” for voice traffic due to high latency may still be “good” for bulk data due to low packet loss.

How SD-WAN Dynamically Load Balances Application Flows

The following diagram shows how SD-WAN load balancing works. Different application data streams can be shared and transported across one or more redundant VPNs. When quality probe metrics for a VPN fall below application performance requirements, or if there is a complete failure of one of the Internet connections, the application flows can automatically recover via the remaining path. Allied Telesis SD-WAN supports both IPv4- and IPv6-based application flows.

Figure 1: SD-WAN load balancing



1. In an SD-WAN setup, the branch sites and/or the central site router has two or more internet connections with different ISPs.
2. Several VPN tunnels are configured between the branch routers and the central router.
3. Each of the tunnels is configured such that it only egresses a specific internet connection. This means that tunnel1 only egresses out eth1, tunnel2 only egresses out eth2, and so on.
4. **Linkmon probes** are configured. The probes are used to determine the quality of links. One probe needs to be configured per tunnel. The probes determine the jitter, latency, packet-loss, and consecutive probe loss of a given link.
5. **Linkmon profiles** are configured. Profiles are a template that describes what metrics are acceptable. They specify how much jitter, latency, packet-loss, and consecutive probe loss are acceptable. You can also define a preferred metric, which is then used when a tie occurs.
6. **Linkmon groups** are defined. A group consists of a list of next-hops, for example, tunnel1, tunnel2, and so on. Each next-hop is known as a member, and each member is tied to a probe. Each group member is assigned an ID which is used when a tie occurs.

7. DPI is configured at the branch routers and the central site to identify application traffic.
8. A PBR linkmon rule is configured. A PBR linkmon rule is configured using the linkmon **ip policy-route** command, and consists of three parts:
   - linkmon profile (as described above)
   - linkmon group (as described above)
   - match criteria. Match criteria describes what traffic is going to be matched by the rule. Traffic is matched based on **application**, **source entity** and **destination entity**. An entity can be a zone, network, or host.
9. Linkmon probes, profiles, and groups need to be configured at the branch sites and at the central site. Each site will independently make their application path forwarding decisions based on their own set of metrics. If rules are not configured at the central site, then return traffic won't be SD-WAN routed. Instead, it will rely on the normal routes available to the device.
10. Client-to-server traffic flows and associated server-to-client reply flows are matched against PBR linkmon rules. Traffic that matches the criteria is forwarded to a linkmon group member. The chosen linkmon group member has metrics discovered by its associated probe better than those defined in the linkmon profile.
11. If multiple linkmon group members (or no linkmon group members) have metrics better than those defined in the linkmon profile, then the group member with the best preferred metric is used. If a preferred metric is not defined, then traffic is sent to the group member with the lowest group member ID that is reachable.

Components of SD-WAN Load Balancing

SD-WAN load balancing has four major components:

- linkmon ip policy-routes
- linkmon groups
- linkmon probes
- linkmon profiles

Each of these components must be configured on each end of the tunnel for a working SD-WAN solution.

Linkmon ip policy-routes

```
ip policy-route [<1-500>]
[match <application_name>] [from <source_entity>]
[to <destination_entity>] linkmon-group <name>
[linkmon-profile <name>]
```

A linkmon ip policy-route, much like a conventional PBR route, is used to redirect traffic which matches the defined match criteria from the standard routing path to a defined nexthop. A linkmon ip policy-route consists of six configurable fields, five of which are optional.

The first field is the **policy-route id**. Much like with firewall and NAT rules, this field describes the order that rules are matched on. For example, if a received packet matches rule 1 and rule 10, then traffic will egress out the nexthop defined by rule 1. The field is optional when entering the **ip policy-route** command. If **policy-route id** is not defined, then the rule is automatically assigned an ID equal to the next multiple of ten. For example, if the highest policy-route ID configured is 12, then a new policy-route would be assigned a value of 20. Policy-routes can be a value of 1 to 500. If a policy-route with an ID of 500 has been created, then automatic assignment of policy-routes will fail. In this case, an ID must be defined.

The following three fields are **match**, **from**, and **to**. They are used to define match criteria for a given PBR rule:

- **match** defines which application you wish to match
- **from** defines which named source entity you wish to match
- **to** defines which named destination entity you wish to match

These fields are optional. If one or more of these fields is not defined, then traffic matches all traffic for that specific field. For example, look at the following policy-route:

```
ip policy-route 5 from LAN linkmon-group GROUP1 linkmon-profile
PROFILE1
```

Because only the **from entity** is defined in the example above, matching traffic from the entity “LAN”, with any destination and associated with any applications, would be policy-routed. If no match criteria such as **application**, **to entity** or **from entity** are defined, then the policy-route will match all traffic received on the device.

The **linkmon-group** field is used to associate the policy-route with a linkmon group. The linkmon group effectively functions as a list of potential nexthops. **linkmon-group** is a mandatory field. For more information, see the section on "Linkmon groups" on page 11.

The **linkmon-profile** field is used to associate the policy-route with a linkmon profile. Quality metrics configured within the linkmon profile are used in choosing the nexthop for traffic matching the defined linkmon PBR policy-route. It defines the latency, jitter, packet-loss, and/or consecutive probe loss metrics acceptable for a given link. This field is optional. If it is not defined, then metric results are not taken into account when determining the nexthop. Instead, the linkmon-group member with the lowest ID which isn't considered down is always used as the nexthop, and so traffic will always be forwarded via the same nexthop path. In this configuration, it behaves identically to a conventional PBR route. For more information, see the section on "Linkmon profiles" on page 15.

Much like with conventional PBR, linkmon PBR rules have to be enabled globally using the **policy-based-routing enable** command.

Linkmon groups

```
linkmon group <name>
member [<member-id>] destination <ip-address>|<interface> probe
<probe-name>
```

A linkmon group is a collection of nexthops that can be used by the associated policy-route. Each linkmon group can have up to eight nexthops, known as **members**. The command to configure a member has three fields, one of which is optional.

The member **id** is a value from 1-128, and is used in some tie-breaker situations when choosing a nexthop. Member **id** is an optional field. If not defined, then the member is automatically assigned an ID the next multiple of 10 above the highest configured member ID. For example, if the highest configured member ID is 25, then the member ID for the next created rule will be 30. If the last configured member ID is 128, then automatic assignment of policy-routes will fail. In this case, an ID must be defined.

The **destination** field defines the nexthop. It can be either an IPv4 or IPv6 address, or a point to point interface (for example, PPP or GRE tunnel). It cannot be configured as a broadcast multi-access interface (for example, VLAN or ETH). If the defined nexthop is in a subnet not directly connected to the device, or is an IP address assigned to an interface on the device, then the member is considered down.

The **probe** field is used to associate the linkmon member to a probe. The probe is used to determine the latency, jitter, packet-loss, and consecutive probe loss of the linkmon member. An IPv6 probe can be used with an IPv4 member and vice versa. If the defined probe does not exist for any

member in the group, then the policy-route associated with the linkmon group functions as if a linkmon profile hasn't been defined. This means metrics are not considered, and traffic is instead routed based on the lowest member ID that is link-up.

Typically, when configuring group members, you need to create a unique probe for each member. For example, for member 1 with a destination of tunnel1, an associated probe will be created that operates over tunnel1, and for member 2 with a destination of tunnel2, an associated probe will be created that operates over tunnel2. For more information, see the section on "Linkmon probes" on page 12.

Linkmon probes

```
linkmon probe name <NAME> [type <icmp-ping|http-get>]
```

Linkmon probe configuration commands and options:

```
destination <A.B.C.D|X:X::X:X|ds-lite|FQDN>
dscp <0-63>
egress interface <interface>
enable
interval <probe-interval>
ip-version <4|6>
sample-size <1-100>
size <64-1500>
source <interface|IPv4-address>
```

A linkmon probe is used to determine latency, jitter, packet-loss, and consecutive probe loss for a given member. The **linkmon probe** command is modal and is used to create and configure the linkmon probe options.

The **name** parameter is used to create a name used to identify the probe.

The **type** parameter lets you specify whether you want to create an ICMP or HTTP probe. By default, an ICMP probe will be created. For more information on creating an HTTP probe, refer to "HTTP probes" on page 13.

Once in configuration mode, the following configuration commands can be used:

The **destination** describes the target of the probe. The destination does not have to match the destination of the linkmon member the probe is associated with.

The destination can be an **IPv4** or **IPv6** address, or alternatively, it can be a fully qualified domain name (FQDN). If an **FQDN** is configured, the device will periodically use DNS to resolve the FQDN to an IP address, and send the ICMP probe to this address. This is useful if you want to probe a particular end-point whose IP address is known to change periodically, for example a remote-office site that may not have a fixed IP address from its ISP. FQDN resolution begins when the probe destination is initially configured, and will recur when the lowest TTL within the returned A/AAAA

records has expired. When **ds-lite** is used, this will direct ICMP probes at the IPv6 address that has been resolved for the DS-Lite AFTR name. The DS-Lite AFTR name can either be statically configured, or learned dynamically via DHCPv6 option 64 (RFC6334).

The **dscp** field allows you to define DSCP field in the IP header of the probes. The DSCP field is optional. By default, probes are sent with a DSCP of “0”.

The **egress** option allows you to configure the interface the probe will be sent out. For IPv4 ICMP probes, only point-to-point interfaces can be specified. IPv6 ICMP probes also support multi-access interfaces.

The **interval** field is used to define the rate at which probes are sent in milliseconds. **interval** is an optional field. By default, probes are sent with an interval of “1000” (one probe per second).

The **ip-version** option is used to specify the IP version used in the probe (IPv4/IPv6).

The **sample-size** field is used to define the sample size used in latency and jitter calculations. **sample-size** is an optional field, with the default value being “5”. The sample size for packet loss is always “100”, and this cannot be changed. The sample size in effect represents the sensitivity of SD-WAN to changes in the health metrics of a link. A low sample size means changes in the metrics will be used to quickly update routing decisions, whereas a larger sample size means updates will occur more slowly. Too small of a sample size can lead to link-flapping, while a large sample size may result in sluggish updates to routing decisions.

The **size** field allows the user to define the packet size of the probes. **size** is an optional field, with probes by default being sent with a size of “100”.

The **source** option allows you specify the source of the probe. Either an interface or an IPv4/IPv6 address can be specified.

The maximum number of link-health monitoring probes that can be configured is restricted to 1000.

Note: linkmon probes are disabled by default, and must be enabled per-probe using the **enable** command.

HTTP probes

```
linkmon probe name <NAME> type http-get
```

Linkmon probe HTTP configuration commands and options:

```
egress interface <interface>
enable
interval <probe-interval>
ip-version <4|6>
url <url>
```

SD-WAN also supports HTTP header request probing. These probes are intended for use when you have redundant ISP links to the Internet. HTTP probing can be used to automatically fail all traffic over to a single ISP if the service becomes unreachable (or has unacceptable latency) via the other ISP.

The linkmon probe configuration mode takes an optional type parameter. By default, if the type is not specified, it will create an ICMP probe. If you set the type parameter to **http-get** it will create an HTTP probe. All configuration commands relating to HTTP probes are under the HTTP probe configuration mode.

To configure an HTTP probe, use the following commands:

```
linkmon probe name MyProbe type http-get
url http://www.alliedtelesis.com/
enable
```

Because HTTP probes are much more resource intensive for an AR-Series firewall to carry out, as well as to avoid the device being identified as performing a TCP DOS or flood attack, the default interval for HTTP probes is 60 seconds rather than 1 second for ICMP. The minimum configurable interval is 30 seconds rather than 100ms. The maximum configurable interval is 3600 seconds rather than 10 seconds.

HTTP probes consist of performing an HTTP HEAD request to the probe destination. This entails performing a DNS lookup (if required), then opening up a TCP connection to the web-server and performing a HEAD request. The durations of each stage of this process are summed together and use as a metric for latency for accessing the website. Jitter is also recorded; however, given the long interval between probes and the fact that HTTP uses TCP, jitter is not a particularly useful metric for HTTP probing. Profiles can still be configured to take jitter into account with HTTP probes if desired.

It is important to configure an HTTP probe using the **url** command, which specifies the website URL that the HTTP HEAD request will be performed against. The URL must use ASCII characters and conform to the URL syntax in RFC 3986, with HTTP or HTTPS protocol at the start and an optional port number on the end, such as :80, :443 or :8080. Some examples of supported URLs:

- http://www.alliedtelesis.com/
- https://www.facebook.com/
- http://intranet.acme.com:8080

HTTP probes are supported for historical metric gathering.

For example, to create a probe named “probe0” and set the URL to “http://www.alliedtelesis.co.nz/” use the following commands:

```
awplus# config terminal
awplus(config)# linkmon probe name probe0 type http-get
awplus(config-linkmon-http-probe)# url http://www.alliedtelesis.co.nz/
```

Linkmon profiles

```
linkmon profile <name>
```

Linkmon profile configuration commands:

```
latency bad-above <1-2000>
latency good-below <1-2000>
jitter bad-above <1-1000>
jitter good-below <1-1000>
pktloss bad-above <0.0-100.0>
pktloss good-below <0.0-100.0>
preference (latency|jitter|pktloss|consecutive-probe-loss|combined)
```

A linkmon profile is used to define what are acceptable metrics for a given linkmon ip policy-route. The **linkmon profile** command is used to create a linkmon profile, or to enter linkmon profile configuration mode, where the profile metrics can be configured. The **name** parameter is used to create a name used to identify the profile.

The quality of a link is based any combination of the metrics: latency, jitter, packet loss, and consecutive probe loss:

- **Latency** is defined as the average round trip time for the last x number of probes, where x is the defined sample size (by default 5).
- **Jitter** is the average difference of latency for the last x number of probes, where x is the defined sample size (5 by default). For example, if the last five probes had latency of 5,10,5,10,5 then the difference between the probes is 5,5,5,5 and so the average jitter is (5+5+5+5)/4=5.
- **Packet loss** is equal to the average number of probes lost for a sample size of 100. A probe is defined to be lost when the reply packet is not received within 2 seconds.
- **Consecutive probe loss** behaves a little differently, and is described in the section "Consecutive probe loss" on page 17.

For each of the metrics there are two definable thresholds, the **bad-above** threshold and the **good-below** threshold. If none of the metrics have **bad-above** thresholds defined, and the linkmon profile is attached to a linkmon PBR rule, then the policy-route behaves like there is no linkmon profile associated with it. This means traffic will be policy-routed to the nexthop with the lowest member ID.

The **preference** command is used to define a preferred metric, and is used in the case of a tie. If we use the metrics of a group and two links are determined to be “good”, we need to select between these links which one is best. For example, if the preference is set to “latency”, then we use that metric to decide which of the two good links is the best link. The “combined” option uses a combination of metrics, and is described in the section "Combined preference" on page 17.

Preference is optional; if no preference is configured, then the group member with the lowest ID will be selected as best when there is a tie amongst links.

Nexthops are chosen according to the following criteria:

- If a nexthop is marked “good”, then it is chosen.
- If multiple links are marked “good”, and a preference is defined, then the link with the best preferred metric is chosen.
- If there is a tie between the best preferred metric, or if a preferred metric isn't defined, then the link with the lowest member ID is chosen.
- If all of the links are marked “bad”, and a preference is defined, then the link with the best preferred metric is chosen.
- If there is a tie between the best preferred metric, or if a preferred metric isn't defined, then the link with the lowest member ID is chosen.
- Once a “bad” link has been chosen, then it will not change until a link becomes “good”.
- If all of the links have been marked “down”, then traffic follows the normal routing path.

If a metric for one of the linkmon members exceeds any of the defined bad-above thresholds, then that link is considered bad. If the last x number of packets have been lost, where x is the defined sample size for latency or jitter (by default 5), then the link is also considered bad. If a good-below value is not defined a link is considered good when the current metric drops below the defined bad-above threshold.

The good-below threshold is designed to add some hysteresis to link changes. For example, picture a link with a linkmon bad-above value of 300 and a good-below value of 200. If the link currently has a latency fluctuating around 305, the link will only be marked good when the latency drops below 200. This means that if the latency is fluctuating around 300 then link-flapping does not occur. For the good-below field to have any functional impact, the bad-above command for the associated metric must also be configured and good-below must be set to a value below the defined bad-above command. If the defined value for the good-below command is greater than the bad-above, then the good-below value is ignored and the system behaves as if only bad-above is configured.

A link is marked “down” if it is administratively or electrically disabled, or if the probes associated with it have 100% packet loss. Due to the large sample size associated with packet loss, there can be severe delays in interfaces using this metric to be marked good.

For example, picture a scenario where the defined bad-above threshold for packet loss is 5% and packet loss on the interface is currently at 99% due to a network outage. After the network outage has been addressed, it will take 95 seconds before the link is marked good. This is because every received probe will decrease the average by 1%, and a probe is sent every second. To mitigate this problem, if a link is at 100% packet loss and receives a reply to one of its sent probes, the average packet loss is reset to 0%.

Consecutive probe loss

Linkmon profile configuration commands:

```
consecutive-probe-loss bad-when <1-100>
consecutive-probe-loss good-when <1-100>
consecutive-probe-loss unreachable-when <1-100>
```

There is also an alternative metric that behaves slightly differently, consecutive probe loss:

- **Consecutive probe loss** measures the number of probes that have been lost in a row.

Profiles can specify how many probes need to be lost before any link associated with that profile and probe will be considered bad or unreachable via the **bad-when** and **unreachable-when** thresholds. The **good-when** threshold applies for when a link is already marked as bad or unreachable. It determines how many consecutive probes must succeed before the link will be considered good.

Consecutive probe loss replicates the functionality (lost pings) provided by the ping-polling feature as part of the link health monitoring feature, for use with link health monitoring triggers. Consecutive probe loss can also be used for standard SD-WAN rules, where it can act as a simpler alternative to the existing packet loss metrics. For more information on link health monitoring triggers, refer to "Link health monitoring triggers" on page 22.

Combined preference

In addition to the basic options of latency, jitter, packet loss, and consecutive probe loss, you can use the “combined” option for the **preference** command. This lets you take a combination of latency, jitter, and packet loss into account when breaking ties to select the best links.

The combined metric is only used for tie-breaking when selecting between links. It produces a score out of 10 for the quality of each link, where a score of 0 represents the best possible link and a score of 10 represents the worst. This metric is calculated using a weighted algorithm of the link’s latency and jitter, with reference to the configured bad-above thresholds in the profile, and the packet loss with a recency weighting applied. If no bad-above threshold is configured for either jitter or latency in the profile, then that metric will not be included when calculating the combined metric. Recency-weighted packet loss is always used as a component for calculating the combined metric.

The combined metric is a more sophisticated way of considering the quality of a link, and could result in more appropriate links being selected to send traffic over. For example, consider a situation where a choice must be made between two links that are both considered good. Link 1 has a latency of 150 and jitter of 70, but link 2 has a latency of 160 and a jitter of 20.

If the preferred metric in this example is latency, then link 1 (with a lower latency of 150) would be selected as the best route in preference to link 2 (which has a latency of 160). By using the combined metric, which takes both latency and jitter together into account, link 2 could be selected as the best available link. While the latency is 10ms worse, the jitter of 20 is much lower than the jitter of 70 that link 1 has.

Similarly, including packet loss as part of the combined metric score could mean that a link with overall low latency, but with periodic packet loss, is avoided in favor of a link with slightly higher latency but zero packet loss.

The current combined metric for linkmon group members can be seen in the output of the **show pbr rules** command.

Linkmon probe-history

```
linkmon probe-history [<1-65535>] probe <NAME> interval <1-2678400>
buckets <1-65535>
```

This command allows you to create a historic data capture for linkmon probes. A custom interval can be configured and a number of buckets, allowing metric data to be flexibly recorded for hours, days, weeks, or months, with resolutions as low as 1 second or as high as days.

Configuring linkmon probe history is optional, and not required for operating SD-WAN. However, it is useful in diagnosing unexpected routing failures.

PBR Routing tables

PBR has a limit of 500 active PBR routing table entries. With linkmon PBR rules, a new PBR route table entry is used for each unique combination of linkmon profile and group.

For example, if the device was configured as follows:

Figure 2: Device configuration using one PBR routing table entry

```
policy-based-routing
 policy-based-routing enable
 ip policy-route 1 match ftp from LAN linkmon-group GROUP1 linkmon-profile
PROFILEONE
 ip policy-route 2 match http from LAN linkmon-group GROUP1 linkmon-profile
PROFILEONE
```

then the configuration would consume one PBR route-table entry. This is because both rules are using GROUP1 and PROFILEONE.

However, if the device was configured like this:

Figure 3: Device configuration using two PBR routing table entries

```
policy-based-routing
 policy-based-routing enable
 ip policy-route 1 match ftp from LAN linkmon-group GROUP1 linkmon-profile
PROFILEONE
 ip policy-route 2 match http from LAN linkmon-group GROUP1 linkmon-profile
PROFILETWO
```

then the configuration would consume two PBR route-tables entries. This is because there are two unique combinations of group and profile; GROUP1 and PROFILEONE, and GROUP1 and PROFILETWO.

If you attempt to configure a rule which requires a 501st PBR route table entry, then the command is rejected.

The current consumption this table can be displayed using the **show pbr rules** command. For more information, refer to "show pbr rules" on page 38.

DPI learning

DPI requires parsing the first few packets of a new packet flow before it can make a decision about which application it is. Initially a flow is marked as “undecided”. As the DPI engine monitors and inspects the flow, it tries to determine which application it is. Once it is sufficiently confident, it will mark the flow as that application.

When DPI learning is enabled, the following occurs:

- The device receives the first packet of a flow between a client and application server.
- DPI is unable to classify the first packet, such as a TCP SYN, as a particular application. It classifies it as “undecided”. Additional packets containing application information need to be received before DPI can make that determination.
- The packet is forwarded down the default path.
- The device receives additional packets associated with this flow.
- The DPI engine classifies this flow as the application, for example Microsoft “office365”.
- Every subsequent packet associated with this particular flow is classified as “office365”. All “office365” application traffic is now redirected, and sent down the high priority path as defined by the PBR rule for the application. The application redirection is achieved by modifying the nexthop IP of the application data flow. It is changed to be the nexthop as configured in the linkmon group associated with the PBR rule.
- Because DPI learning is enabled, the DPI engine associates the destination IP, destination port, and IP protocol number of this flow with the application “office365”. It stores this information within the DPI learning cache.
- The device receives the first packet of a new flow from a different client to same application server.
- Because the new flow has the same destination IP, destination port, and IP protocol number as the DPI cache entry for the original flow, the DPI engine classifies the new flow as “office365” from the initial TCP SYN packet.
- All packets associated with the new flow are classified as “office365”, and are sent down the high priority path.

This works well when the application traffic is transported via redundant VPNs between sites. The application traffic can switch paths mid-flow once it is identified by DPI, without interrupting the client-to-server communications. For an example of this configuration, refer to "Configuration Examples - Application Aware Routing Via Redundant VPNs" on page 24.

However, certain SD-WAN configurations need to be able to apply their policies right from the first packet in a flow, and are not able to change their decision. In the example above, the initial flow changes from the low priority path to high priority path mid-flow. If intermediate NAT is being used on either of these paths, then this will cause a communication failure between the client and server. This is because the source IP of the traffic to the server will change via NAT once the flow changes to the high priority path and egresses a different interface. For an example of this configuration, refer to "Configuration Example - Proxy Server Bypass" on page 34.

This is typically a problem if using SD-WAN to control application paths directly to the Internet, and the router or an intermediate device is using NAT. The **application-decision once-only** command is used resolve this. If this command has been enabled in policy-based-routing configuration mode, then all traffic associated with the initial flow will continue to be classified as “undecided”. Only subsequent flows will be classified as “office365”.

DPI learning builds up a cache of the DPI decisions made for server IP address, destination port, and IP protocol number combinations. This allows future flows to that address and port combination to be matched to the associated application from the first packet. Meanwhile, DPI continues to monitor flows, and updates the cache if it determines that the application associated with that address and port combination has changed. This ensures that future flows are marked with the most appropriate application right from the first packet.

DPI learning configuration example

This example describes how to configure DPI learning.

1) Enter the DPI mode.

```
awplus#configure terminal
awplus(config)#dpi
```

2) Enable the DPI learning feature.

```
awplus(config-dpi)#learning
```

3) Verify the DPI learning configuration.

Figure 4: Example output for the show dpi command with DPI learning enabled

```
awplus#show dpi
Status: disabled
Provider: not set
Mode:   learning
Resource version: not set
Resource update interval: 1 hour
```

By default, DPI learning has a cache size of 10,000 entries. To set it to a different amount, use the following command:

```
awplus(config-dpi)#learning cache <size>
```

The cache can be set to any value between 50 and 16000.

4) Set the DPI provider and enable DPI.

```
awplus(config-dpi)#provider procera
awplus(config-dpi)#enable
```

5) Verify the DPI learning configuration.

Figure 5: Example output for the show dpi command with DPI learning running

```
awplus#show dpi
Status: running
Provider: procera
Mode:   learning
Resource version:          dpi_procera_app_db_v52
Resource update interval: 1 hour
```

Note: DPI learning is supported for both Procera and built-in providers.

Tunnel security reprocessing

When application traffic is transported within a VPN, it is typically encrypted or encapsulated within VPN headers. This means when it arrives at its destination, the encapsulated application traffic cannot be examined by DPI, particularly if it is encrypted. DPI will identify the traffic by its VPN headers.

For example, the VPN traffic will be identified as ‘IPSEC on ingress from the WAN interface’. Tunnel security reprocessing allows packets that arrive from tunnels that terminate on the device to be re-inspected after decryption. This allows DPI to inspect the contents of those decrypted packets, and identify the embedded application traffic correctly. It can then route the application traffic according to any PBR rules.

To enable tunnel security reprocessing, use the following command:

```
tunnel security-reprocessing
```

For an example of this configuration, refer to "Configuration Examples - Application Aware Routing Via Redundant VPNs" on page 24. For more information about tunnel security-reprocessing, refer to the IPSec Feature Overview and Configuration Guide.

Caution Tunnel security reprocessing increases the load on your device and reduces throughput. This is because VPN traffic is processed twice via DPI:

- As the traffic ingresses the physical interface, it is examined by DPI and identified based on the outer VPN encapsulation. For example, IPSEC encrypted traffic is identified as IPSEC by the DPI engine.
- The traffic is de-encrypted and the outer VPN encapsulation is removed. When tunnel security reprocessing is enabled, the decapsulated data stream is passed through the DPI engine a second time.

Link health monitoring triggers

You can configure triggers that respond to link health monitoring events, produced by a combination of a link health monitoring probe and profile. This configuration acts as a more powerful alternative to ping-poll triggers, since you can use the jitter and latency results from the probe to determine if the path to the destination IP address is bad or good, in addition to determining if the destination is up or down.

This is useful if you want the monitor the health of the link and use trigger scripts to alter the device configuration in response to the link health changing, for example to alter the administrative distance of default routes if a destination becomes unreachable. These triggers require a linkmon probe to be configured, as well as a profile that can be used to judge whether the probe results are good, bad, or if the destination is unreachable.

For more information about ping-poll triggers, refer to the Ping Polling Feature Overview and Configuration Guide.

Link health monitoring trigger configuration

The first example below shows how to create a ping-poll trigger that will be triggered when the link health changes.

Figure 6: Example ping-poll trigger configuration

```
ping-poll 1
 ip 192.168.1.1
 up-count 3
 fail-count 4
 active
!
trigger 1
 type ping-poll 1 down
```

The second example shows how to create the equivalent configuration using link health monitoring probes and profiles, and a link health monitoring trigger.

Figure 7: Example link health monitoring trigger configuration

```
linkmon probe name MyProbe type icmp-ping
 destination 192.168.1.1
 interval 1000
 enable
!
linkmon profile MyProfile
 consecutive-probe-loss good-when 3
 consecutive-probe-loss unreachable-when 4
trigger 2
 type linkmon-probe MyProbe MyProfile unreachable
```

Application-based aggregation via redundant VPNs

Application-based aggregation allows you to load-balance traffic that matches a PBR rule across multiple SD-WAN link members at a time. This may be useful if you have multiple redundant VPN paths to your destination. This allows you to utilize as much bandwidth of those paths as possible, while still being able to automatically redirect traffic away from links that are determined to be “bad”. Aggregation is configured on a linkmon group, or directly on a conventional PBR rule.

Aggregation is achieved using a hashing algorithm on a per-flow basis for each application. For IPv4 packets, the hashing algorithm uses the source and destination IP addresses. For IPv6, it also includes the source and destination port, GRE key (if applicable), protocol, and an internal flow label attached to each flow. Because the hashing applies per flow, it means two users whose traffic is matched by a PBR rule may have the traffic associated with their flows sent over different paths, but all packets within the flow will take the same path. This avoids packet re-sequencing problems at the traffic's destination, and allows flows for the same application to be aggregated over both paths.

When a linkmon group is configured for aggregation, any preferred metric configured in the profile associated with that rule will be ignored. Instead, traffic that is directed to that group's nexthops will be hashed per flow between all links within the group that are currently marked as “good” by the associated linkmon profile. If there are no “good” links, then all traffic will be sent over only the single best “bad” link available. For conventional PBR rules, the traffic will be balanced across all links that are up.

Aggregation works in conjunction with the virtual-bandwidth commands provided by the Traffic Control feature. This means that multiple tunnels can be used to load-balance traffic, but if they have largely different speeds, such as a 100Mb link and a 10Mb link, virtual-bandwidth can be used to ensure that the majority of traffic is sent over the faster link, but the slower link can also be used to transmit data.

Configuration

To use aggregation on a conventional PBR rule, use the following commands with the **load-balance** parameter:

```
awplus(config)# policy-based-routing
awplus(config-pbr)# ip[v6] policy-route match https from private to
public nexthop tunnel10 tunnel20 load-balance
```

To enable aggregation for a particular linkmon group, use the following commands:

```
awplus(config)# linkmon group MyGroup
awplus(config-linkmon-group)# load-balance
awplus(config-linkmon-group)# member 1 destination tunnel10 probe
probe10
awplus(config-linkmon-group)# member 2 destination tunnel20 probe
probe20
```

(Optional) To assign a virtual-bandwidth to each tunnel, use the following commands:

```
awplus(config)# traffic-control
```

```
awplus(config-tc)# interface tunnel10 virtual-bandwidth 10kbit
awplus(config-tc)# interface tunnel20 virtual-bandwidth 100kbit
awplus(config-tc)# traffic-control enable
```

Configuration Examples - Application Aware Routing Via Redundant VPNs

This example configuration ensures that the real-time protocol SIP is sent between the two remote sites, BRANCH and CENTRAL_SITE, while ensuring a level of quality on the link. Real-time protocols are used when sending time-sensitive traffic. Some notable examples are:

- H323 and SIP, which are used for voice and video traffic.
- RDP and VNC, which are used for remote access to a workstation.

By their nature, they are highly sensitive to adverse network conditions such as high latency and jitter. For these applications to be used effectively, the connection they are sent over has to have a fairly high committed information rate (CIR) over a link.

VPN technologies come in two major types:

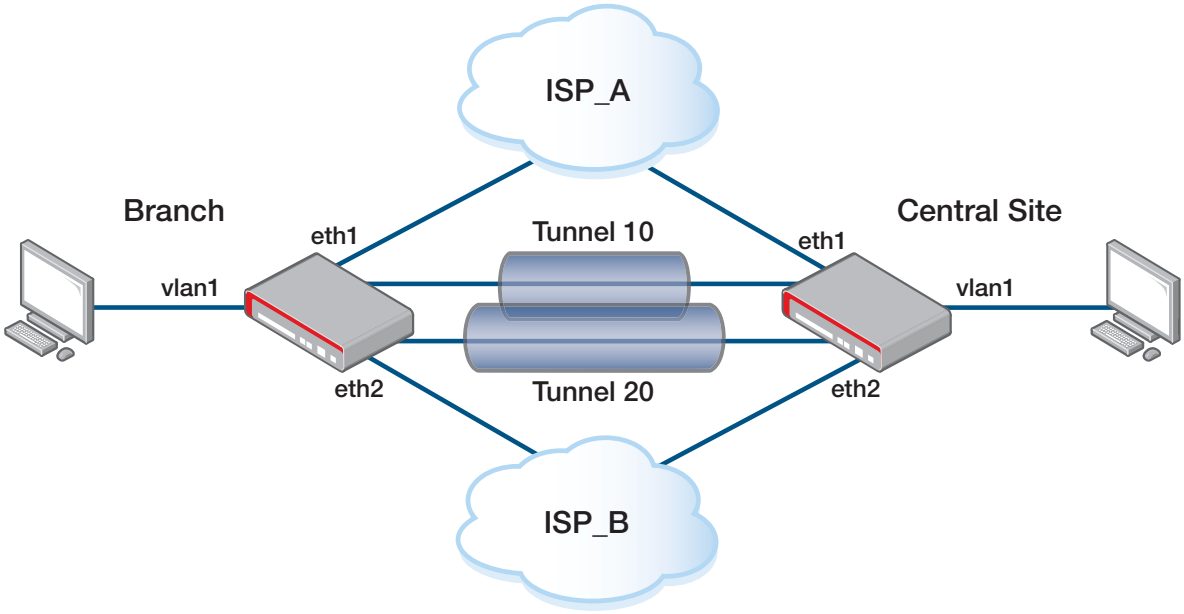
- Dedicated links purchased from an ISP. These are extremely expensive, but have a guaranteed CIR or bandwidth guarantees.
- Using a tunnelling protocol across conventional internet links. This is highly cost-effective; however, because it is dependent on conventional Internet connections, the performance can and does vary dramatically.

This SD-WAN example ensures a degree of link quality over cheap tunnel-based VPNs. It does this by having two tunnels over two different Internet connections offered by two different ISPs. A probing mechanism is used over each of these tunnels to determine the latency, jitter, packet-loss, and consecutive probe loss of each tunnel. SIP-related traffic will then be policy-routed onto a tunnel which meets the performance requirements described in the linkmon profile “PROFILE1” (unless neither tunnel meets the described minimum requirements).

The configuration supplied only matches SIP-related traffic. It does this using DPI, which is able to determine applications based on layer 7 information. This allows for more granular control over what traffic is or isn't redirected. Only SIP connections initiated at BRANCH are policy-routed. The policy-routes at the CENTRAL_SITE are used to ensure return traffic takes the correct path.

IPv4 and IPv6 SD-WAN configuration examples for this solution are below.

Figure 8: An example of policy-based routing configuration



IPv4 example

Figure 9: Example of branch configuration - IPv4

```
!
hostname BRANCH
!
zone CENTRAL_WAN
 network ETH1
  ip subnet 192.0.2.2/32 interface eth1
 network ETH2
  ip subnet 192.0.2.6/32 interface eth2
!
zone LAN
 network TUNNEL
  ip subnet 192.168.10.0/30 interface tunnel10
  ip subnet 192.168.20.0/30 interface tunnel20
 network VLAN1
  ip subnet 172.16.10.0/24 interface vlan1
 network CENTRAL_LAN
  ip subnet 172.16.1.0/24
!

zone WAN
 network ETH
  ip subnet 0.0.0.0/0 interface eth1
  ip subnet 0.0.0.0/0 interface eth2
  host HOST
   ip address 198.51.100.2
   ip address 198.51.100.6
!
application esp
 protocol 50
!
application isakmp
 protocol udp
 dport 500
!
```

```
firewall
 rule 10 permit any from LAN to LAN
 rule 20 permit any from LAN to WAN
 rule 30 permit isakmp from CENTRAL_WAN to WAN
 rule 40 permit esp from CENTRAL_WAN to WAN
 rule 50 permit any from WAN.ETH.HOST to WAN.ETH
 protect
!
nat
 rule 10 masq any from LAN.VLAN1 to WAN
 enable
!
dpi
 provider procera
 enable
!
crypto isakmp key <samplekey1> hostname TUNNEL10
crypto isakmp key <samplekey2> hostname TUNNEL20
!

policy-based-routing    ←policy-route describes which traffic to match and where to route it.
 ip policy-route 10 match sip from LAN.VLAN1 linkmon-group GROUP1 linkmon-
profile PROFILE1
 ip policy-route 20 match rtp from LAN.VLAN1 linkmon-group GROUP1 linkmon-
profile PROFILE1
 ip policy-route 30 match rtcp from LAN.VLAN1 linkmon-group GROUP1 linkmon-
profile PROFILE1
 policy-based-routing enable
!
linkmon probe name PROBE1    ←defines a linkmon probe used to determine the quality of a link.
 destination 192.168.10.1
 enable
!
linkmon probe name PROBE2
 destination 192.168.20.1
 enable
!

linkmon group GROUP1    ←list of possible nexthops as known linkmon members. Each member is
                          associated with a probe.
 member 10 destination tunnel20 probe PROBE2
 member 20 destination tunnel10 probe PROBE1
!
linkmon profile PROFILE1    ←defines the acceptable metrics for a given link.
 latency bad-above 150
 latency good-below 100
 jitter bad-above 40
 jitter good-below 20
 consecutive-probe-loss good-when 3
 consecutive-probe-loss bad-when 4
 consecutive-probe-loss unreachable-when 7
!
ip name-server 8.8.8.8
ip domain-lookup
!

tunnel security-reprocessing    ←allows DPI to re-inspect traffic after VPN decapsulation.
!
interface eth1
 ip address 198.51.100.2/30
!
interface eth2
 ip address 198.51.100.6/30
!
interface vlan1
 ip address 172.16.10.1/24
!
```

```
interface tunnel10
 tunnel source eth1
 tunnel destination 192.0.2.2
 tunnel local name TUNNEL10
 tunnel remote name TUNNEL10
 tunnel protection ipsec
 tunnel mode ipsec ipv4
 ip address 192.168.10.2/30
!

interface tunnel20
 tunnel source eth2
 tunnel destination 192.0.2.6
 tunnel local name TUNNEL20
 tunnel remote name TUNNEL20
 tunnel protection ipsec
 tunnel mode ipsec ipv4
 ip address 192.168.20.2/30
!

ip route 0.0.0.0/0 198.51.100.1
ip route 0.0.0.0/0 198.51.100.5 254
ip route 172.16.1.0/24 tunnel10
ip route 192.0.2.0/30 198.51.100.1
ip route 192.0.2.0/30 Null 254
ip route 192.0.2.4/30 198.51.100.5
ip route 192.0.2.4/30 Null 254
!
```

Figure 10: Example of central site configuration - IPv4

```
!
hostname CENTRAL_SITE
!
zone BRANCH_WAN
 network ETH1
  ip subnet 198.51.100.2/32 interface eth1
 network ETH2
  ip subnet 198.51.100.6/32 interface eth2
!

zone LAN
 network TUNNEL
  ip subnet 192.168.10.0/30 interface tunnel10
  ip subnet 192.168.20.0/30 interface tunnel20
 network VLAN1
  ip subnet 172.16.1.0/24 interface vlan1
 network BRANCH_LAN
  ip subnet 172.16.10.0/24
!

zone WAN
 network ETH
  ip subnet 0.0.0.0/0 interface eth1
  ip subnet 0.0.0.0/0 interface eth2
  host HOST
   ip address 192.0.2.2
   ip address 192.0.2.6
!
application esp
 protocol 50
!
application isakmp
 protocol udp
 dport 500
!
```

```
firewall
 rule 10 permit any from LAN to LAN
 rule 20 permit any from LAN to WAN
 rule 30 permit isakmp from BRANCH_WAN to WAN
 rule 40 permit esp from BRANCH_WAN to WAN
 rule 50 permit any from WAN.ETH.HOST to WAN.ETH
 protect
!

nat
 rule 10 masq any from LAN.VLAN1 to WAN
 enable
!
dpi
 provider procera
 enable
!
crypto isakmp key <samplekey1> hostname TUNNEL10
crypto isakmp key <samplekey2> hostname TUNNEL20
!

policy-based-routing
 ip policy-route 10 match sip to LAN.BRANCH_LAN linkmon-group GROUP1 linkmon-
profile PROFILE1
 ip policy-route 20 match rtp to LAN.BRANCH_LAN linkmon-group GROUP1 linkmon-
profile PROFILE1
 ip policy-route 30 match rtcp to LAN.BRANCH_LAN linkmon-group GROUP1 linkmon-
profile PROFILE1
 policy-based-routing enable
!
linkmon probe name PROBE1
 destination 192.168.10.2
 enable
!
linkmon probe name PROBE2
 destination 192.168.20.2
 enable
!

linkmon group GROUP1
 member 10 destination tunnel20 probe PROBE2
 member 20 destination tunnel10 probe PROBE1
!
linkmon profile PROFILE1
 latency bad-above 150
 latency good-below 100
 jitter bad-above 40
 jitter good-below 20
 consecutive-probe-loss good-when 3
 consecutive-probe-loss bad-when 4
 consecutive-probe-loss unreachable-when 7
!
ip name-server 8.8.8.8
ip domain-lookup
!

tunnel security-reprocessing
!
interface eth1
 ip address 192.0.2.2/30
!
interface eth2
 ip address 192.0.2.6/30
!
interface vlan1
 ip address 172.16.1.1/24
!
```

```
interface tunnel10
 tunnel source eth1
 tunnel destination 198.51.100.2
 tunnel local name TUNNEL10
 tunnel remote name TUNNEL10
 tunnel protection ipsec
 tunnel mode ipsec ipv4
 ip address 192.168.10.1/30
!
interface tunnel20
 tunnel source eth2
 tunnel destination 198.51.100.6
 tunnel local name TUNNEL20
 tunnel remote name TUNNEL20
 tunnel protection ipsec
 tunnel mode ipsec ipv4
 ip address 192.168.20.1/30
!

ip route 0.0.0.0/0 192.0.2.1
ip route 0.0.0.0/0 192.0.2.5 254
ip route 172.16.10.0/24 tunnel10
ip route 198.51.100.0/30 192.0.2.1
ip route 198.51.100.0/30 Null 254
ip route 198.51.100.4/30 192.0.2.5
ip route 198.51.100.4/30 Null 254
!
```

IPv6 example

Figure 11: Example of branch configuration - IPv6

```
!
hostname BRANCH
!
zone CENTRAL_WAN
 network ETH1
  ipv6 subnet 2001:db8:1:1::2/128 interface eth1
 network ETH2
  ipv6 subnet 2001:db8:1:2::2/128 interface eth2
!

zone LAN
 network CENTRAL_LAN
  ipv6 subnet 2001:db8:1:3::/64
 network TUNNEL10
  ipv6 subnet fd00:10::/64 interface tunnel10
 network TUNNEL20
  ipv6 subnet fd00:20::/64 interface tunnel20
 network VLAN1
  ipv6 subnet 2001:db8:2:3::/64 interface vlan1
!
zone WAN
 network ETH
  ipv6 subnet ::/0 interface eth1
  ipv6 subnet ::/0 interface eth2
  host HOST
   ipv6 address 2001:db8:2:1::2
   ipv6 address 2001:db8:2:2::2
!

application esp
 protocol 50
!
application isakmp
 protocol udp
 dport 500
!
```

```
firewall
 rule 10 permit any from LAN to LAN
 rule 20 permit any from LAN to WAN
 rule 30 permit isakmp from CENTRAL_WAN to WAN
 rule 40 permit esp from CENTRAL_WAN to WAN
 rule 50 permit any from WAN.ETH.HOST to WAN.ETH
 protect
!

dpi
 provider procera
 enable
!
crypto isakmp key <samplekey1> hostname TUNNEL10
crypto isakmp key <samplekey2> hostname TUNNEL20
!

policy-based-routing    ←policy-route describes which traffic to match and where to route it.
 ipv6 policy-route 10 match sip from LAN.VLAN1 linkmon-group GROUP1 linkmon-
profile PROFILE1
 ipv6 policy-route 20 match rtp from LAN.VLAN1 linkmon-group GROUP1 linkmon-
profile PROFILE1
 ipv6 policy-route 30 match rtcp from LAN.VLAN1 linkmon-group GROUP1 linkmon-
profile PROFILE1
 policy-based-routing enable
!

linkmon probe name PROBE1    ←defines a linkmon probe used to determine the quality of a link.
 ip-version 6
 destination fd00:10::1
 enable
!
linkmon probe name PROBE2
 ip-version 6
 destination fd00:20::1
 enable
!

linkmon group GROUP1    ←list of possible nexthops as known linkmon members. Each member is
                          associated with a probe.
 member 10 destination tunnel20 probe PROBE2
 member 20 destination tunnel10 probe PROBE1
!
linkmon profile PROFILE1    ←defines the acceptable metrics for a given link.
 latency bad-above 150
 latency good-below 100
 jitter bad-above 40
 jitter good-below 20
 consecutive-probe-loss good-when 3
 consecutive-probe-loss bad-when 4
 consecutive-probe-loss unreachable-when 7
!
ip name-server 2001:4860:4860::8888
ip domain-lookup
!

tunnel security-reprocessing    ←allows DPI to re-inspect traffic after VPN decapsulation.
!
interface eth1
 ipv6 address 2001:db8:2:1::2/64
!
interface eth2
 ipv6 address 2001:db8:2:2::2/64
!
interface vlan1
 ipv6 address 2001:db8:2:3::1/64
!
```

```
interface tunnel10
 tunnel source eth1
 tunnel destination 2001:db8:1:1::2
 tunnel local name TUNNEL10
 tunnel remote name TUNNEL10
 tunnel protection ipsec
 tunnel mode ipsec ipv6
 ipv6 address fd00:10::2/64
!
interface tunnel20
 tunnel source eth2
 tunnel destination 2001:db8:1:2::2
 tunnel local name TUNNEL20
 tunnel remote name TUNNEL20
 tunnel protection ipsec
 tunnel mode ipsec ipv6
 ipv6 address fd00:20::2/64
!

ipv6 forwarding
!
ipv6 route ::/0 2001:db8:2:1::1
ipv6 route ::/0 2001:db8:2:2::1 254
ipv6 route 2001:db8:1:1::2/128 2001:db8:2:1::1
ipv6 route 2001:db8:1:1::2/128 Null 254
ipv6 route 2001:db8:1:2::2/128 Null 254
ipv6 route 2001:db8:1:2::2/128 2001:db8:2:2::1
ipv6 route 2001:db8:1:3::/64 tunnel10
!
```

Figure 12: Example of central site configuration - IPv6

```
!
hostname CENTRAL_SITE
!
zone BRANCH_WAN
 network ETH1
  ipv6 subnet 2001:db8:2:1::2/128 interface eth1
 network ETH2
  ipv6 subnet 2001:db8:2:2::2/128 interface eth2
!

zone LAN
 network BRANCH_LAN
  ipv6 subnet 2001:db8:2:3::/64
 network LAN
  ipv6 subnet 2001:db8:1:3::/64 interface vlan1
 network TUNNEL10
  ipv6 subnet fd00:10::/64 interface tunnel10
 network TUNNEL20
  ipv6 subnet fd00:20::/64 interface tunnel20
!

zone WAN
 network ETH
  ipv6 subnet ::/0 interface eth1
  ipv6 subnet ::/0 interface eth2
  host HOST
   ipv6 address 2001:db8:1:1::2
   ipv6 address 2001:db8:1:2::2
!
application esp
 protocol 50
!
application isakmp
 protocol udp
 dport 500
!
```

```
firewall
 rule 10 permit any from LAN to LAN
 rule 20 permit any from LAN to WAN
 rule 30 permit isakmp from BRANCH_WAN to WAN
 rule 40 permit esp from BRANCH_WAN to WAN
 rule 50 permit any from WAN.ETH.HOST to WAN.ETH
 protect
!

dpi
 provider procera
 enable
!
crypto isakmp key <samplekey1> hostname TUNNEL10
crypto isakmp key <samplekey2> hostname TUNNEL20
!

policy-based-routing
 ipv6 policy-route 10 match sip to LAN.BRANCH_LAN linkmon-group GROUP1 linkmon-
profile PROFILE1
 ipv6 policy-route 20 match rtp to LAN.BRANCH_LAN linkmon-group GROUP1 linkmon-
profile PROFILE1
 ipv6 policy-route 30 match rtcp to LAN.BRANCH_LAN linkmon-group GROUP1 linkmon-
profile PROFILE1
 policy-based-routing enable
!
linkmon probe name PROBE1
 ip-version 6
 destination fd00:10::2
 enable
!
linkmon probe name PROBE2
 ip-version 6
 destination fd00:20::2
 enable
!

linkmon group GROUP1
 member 10 destination tunnel20 probe PROBE2
 member 20 destination tunnel10 probe PROBE1
!
linkmon profile PROFILE1
 latency bad-above 150
 latency good-below 100
 jitter bad-above 40
 jitter good-below 20
 consecutive-probe-loss good-when 3
 consecutive-probe-loss bad-when 4
 consecutive-probe-loss unreachable-when 7
!
ip name-server 2001:4860:4860::8888
ip domain-lookup
!

tunnel security-reprocessing
!
interface eth1
 ipv6 address 2001:db8:1:1::2/64
!
interface eth2
 ipv6 address 2001:db8:1:2::2/64
!
interface vlan1
 ipv6 address 2001:db8:1:3::1/64
!
```

```
interface tunnel10
 tunnel source eth1
 tunnel destination 2001:db8:2:1::2
 tunnel local name TUNNEL10
 tunnel remote name TUNNEL10
 tunnel protection ipsec
 tunnel mode ipsec ipv6
 ipv6 address fd00:10::1/64
!
interface tunnel20
 tunnel source eth2
 tunnel destination 2001:db8:2:2::2
 tunnel local name TUNNEL20
 tunnel remote name TUNNEL20
 tunnel protection ipsec
 tunnel mode ipsec ipv6
 ipv6 address fd00:20::1/64
!
ipv6 forwarding
!
ipv6 route ::/0 2001:db8:1:1::1
ipv6 route ::/0 2001:db8:1:2::1 254
ipv6 route 2001:db8:2:1::2/128 2001:db8:1:1::1
ipv6 route 2001:db8:2:1::2/128 Null 254
ipv6 route 2001:db8:2:2::2/128 2001:db8:1:2::1
ipv6 route 2001:db8:2:2::2/128 Null 254
ipv6 route 2001:db8:2:3::/64 tunnel10
!
```

Configuration Example - Proxy Server Bypass

In this example, a proxy server is being used for HTTP and HTTPS traffic. The proxy server is initiating proxied connections using the IP on its WAN interface (172.16.3.2), rather than spoofing the source IP of the host that initiated the HTTP/HTTPS connection. Policy rules are configured on the internal router to direct all web traffic via the proxy server (the green arrows). However the proxy server is being overwhelmed when there are over-the-Internet updates for Office365 from the clients. To help reduce some of the load to the proxy server, policy-routing is used to redirect all Microsoft Office 365 traffic directly to the Firewall edge router (the orange arrows), rather than through the proxy. All other traffic for which there are no policy routes are routed based on the static default route.

Note If the proxy server is not transparent, client workstations will typically be configured to send their application traffic to the destination IP address of the proxy server, not to the server located on the Internet directly. This means DPI cannot learn the destination IP addresses of the application server located in the Internet. In this situation, you should install proxy auto-configuration (PAC) files into the workstations. This ensures select application traffic can bypass the proxy and be sent directly to the application server destination IP address on the Internet.

Figure 13: Proxy server bypass diagram

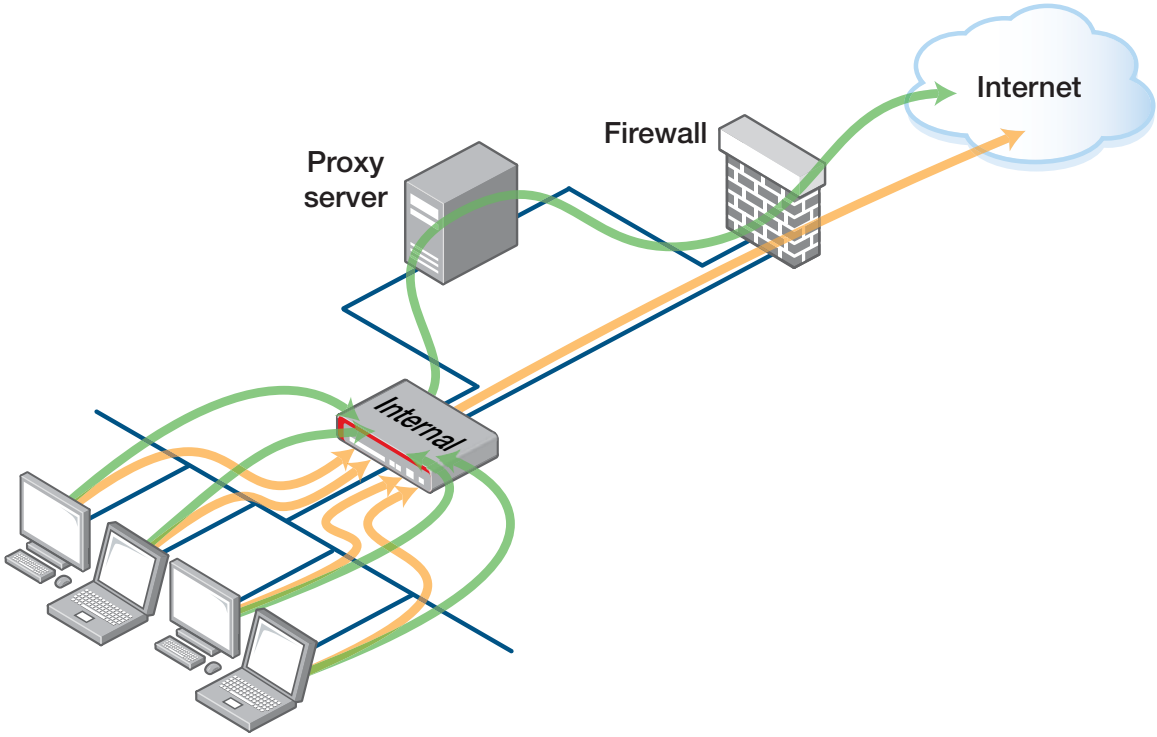


Figure 14: Proxy server bypass logical network diagram

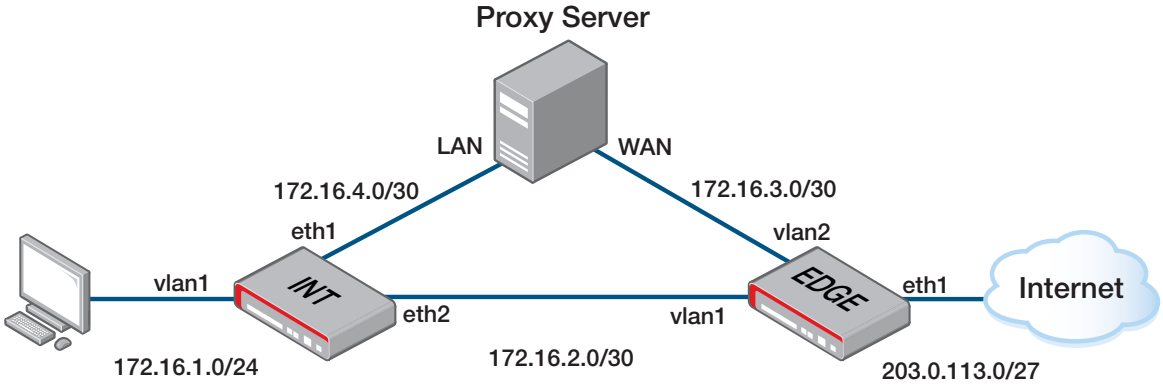


Figure 15: Internal configuration

```
!
hostname INT
!
zone ANY
 network ANY
  ip subnet 0.0.0.0/0
!
zone LAN
 network LAN
  ip subnet 172.16.1.0/24
!

application http
 protocol tcp
 dport 80
!
application https
 protocol tcp
 dport 443
!
dpi
 provider procera
 learning
 enable
!

policy-based-routing
 application-decision once-only
 policy-based-routing enable
 ip policy-route 10 match microsoft from LAN nexthop 172.16.2.1
 ip policy-route 20 match office from LAN nexthop 172.16.2.1
 ip policy-route 30 match http from LAN nexthop 172.16.4.2
 ip policy-route 40 match https from LAN nexthop 172.16.4.2
!

ip name-server 10.36.250.13
ip name-server 8.8.8.8
ip domain-lookup
!
ip dhcp pool LAN
 network 172.16.1.0 255.255.255.0
 range 172.16.1.10 172.16.1.100
 dns-server 8.8.8.8
 default-router 172.16.1.1
!
interface eth1
 ip address 172.16.4.1/30
!
interface eth2
 ip address 172.16.2.2/30
!
```

```
interface vlan1
 ip address 172.16.1.1/24
!
ip route 0.0.0.0/0 172.16.2.1
!
```

Figure 16: Edge configuration

```
!
hostname EDGE
!
zone LAN
 network LAN
  ip subnet 172.16.1.0/24
 network MANAGEMENT
  ip subnet 172.16.2.0/24
 network PROXY
  ip subnet 172.16.3.0/24
!

zone WAN
 network ETH1
  ip subnet 0.0.0.0/0 interface eth1
!
firewall
 rule 10 permit any from LAN to LAN
 rule 20 permit any from LAN to WAN
 rule 30 permit any from WAN.ETH1 to WAN
 protect
!
nat
 rule 10 masq any from LAN to WAN
 enable
!

ip domain-lookup
!
vlan database
 vlan 2 state enable
!
interface port1.0.2
 switchport access vlan 2
!

interface eth1
 ip address 203.0.113.2/27
!
interface vlan1
 ip address 172.16.2.1/30
!
interface vlan2
 ip address 172.16.3.1/30
!
ip route 172.16.1.0/24 172.16.2.2
ip route 0.0.0.0/0 203.0.113.1
!
```

Other Configuration Options

Linkmon probe-history configuration

Linkmon probe histories can be use to store a history of probe metric information.

The following commands show how to create three linkmon probes. For each probe, we will use an interval of 5 seconds, and a sample of 100 buckets.

```
awplus#conf t
Enter configuration commands, one per line. End with CNTL/Z.
awplus(config)#linkmon probe-history 10 probe PROBE1 interval 5 buckets
100
awplus(config)#linkmon probe-history 11 probe PROBE2 interval 5 buckets
100
awplus(config)#linkmon probe-history 12 probe PROBE3 interval 5 buckets
100
```

The CLI **show linkmon probe-history** command output below shows the Minimum, Maximum, and Average Latency, as well as Jitter and Packet Loss. The results for this example are 1 sample taken every 5 seconds, and the most recent 100 samples are used. This means we can see the results over a 500 second period.

From this, we can see what the metrics are like on a link over that time period:

- PROBE1 is experiencing high latency and jitter.
- PROBE2 is experiencing high but consistent latency.
- PROBE3 is seeing ~18% packet loss.

Figure 17: Output from the show linkmon probe-history command

```
awplus#show linkmon probe-history
-------------------------------------------------------------------------------
ID     Interval (s)      Buckets   Latency (ms): Min           Max           Avg
Probe                              Jitter (ms): Min           Max           Avg
                                       Packets:  Tx            Rx      Loss (%)
-------------------------------------------------------------------------------
10                5      100/100                447           551           497
PROBE1                                          26           137            66
                                               499           499          0.00
11                5      100/100                 99           100            99
PROBE2                                           0             1             0
                                               500           500          0.00
12                5      100/100                  0             0             0
PROBE3                                           0             0             0
                                               499           408         18.24
```

Null interfaces

The pseudo-interface Null has been added to PBR to drop packets. Null acts as an always up but bad interface. It can be used as a last choice nexthop when all other nexthops are unavailable so that packets are dropped instead of following conventional routing. Null can be added as the nexthop in a PBR rule, or as a group member in a Link Health Monitoring group where no probe is required.

Figure 18: Configuration using Null pseudo-interface

```
awplus(config-pbr)#ip policy-route match http nexthop tunnel1 tunnel2 Null

awplus(config)#linkmon group TUNNELS_NULL
awplus(config-linkmon-group)#member 10 destination tunnel1 probe TUNNEL1
awplus(config-linkmon-group)#member 20 destination tunnel2 probe TUNNEL2
awplus(config-linkmon-group)#member 30 destination Null
awplus(config-linkmon-group)#^D
awplus(config)#policy-based-routing
awplus(config-pbr)#ip policy-route match http linkmon-group TUNNELS_NULL
```

Recommended linkmon profile settings

SIP and H323

The amount of latency, jitter, and packet loss that is acceptable for voice traffic on a given link is dependent on a range of factors. As a general rule, latency should not exceed 150ms, and jitter should not exceed 40ms.

Note Most sources recommend that voice traffic packet loss should not exceed 1%. However, configuring a packet loss metric of 1% or less is not recommended, as this will likely lead to flapping. We recommend setting the packet loss metric to 5%.

RDP

Acceptable network conditions are largely dependent on the workload you are supporting. A latency of under 300ms is usable in most applications. Jitter has a negligible effect on RDP. RDP is highly susceptible to packet loss, so this should be configured relatively low.

Show Commands

show pbr rules

Command

```
show pbr rules [profile <NAME>]|[group <NAME>]|[<rule ID>]
```

This command is used to display all configuration and status related to PBR rules.

To show all configuration for PBR rules, use the following command:

```
awplus#show pbr rules
```

To show the configuration for a specific PBR rule, use the following command:

```
awplus#show pbr rules 1
```

To show the configuration for a specific linkmon profile named “DELAY”, use the following command:

```
awplus#show pbr rules profile DELAY
```

To show the configuration for a specific linkmon group named “GROUP1”, use the following command:

```
awplus#show pbr rules group GROUP1
```

Figure 19: Example output for the show pbr rules command

```
awplus#show pbr rules
Statistics:
------------------------------------------------------------------------
Route table usage: 1/62
Total number of configured PBR-rules = 1
------------------------------------------------------------------------

PBR-Rule 5
------------------------------------------------------------------------


Active:                Yes
Match:                 sip
From:                  LAN
To:                    any
Profile:               PROFILE1
  latency bad above                : 300 ms
  latency good below               : - ms
  jitter bad above                 : 40 ms
  jitter good below                : - ms
  pktloss bad above                : 40.0 %
  pktloss good below               : - %
  Consecutive success good when    : 3
  Consecutive loss bad when        : 4
  Consecutive loss unreachable when : 7
  preference:          least latency


Group:                   GROUP1
  Member:                10
    Next-hop:            172.16.10.1
    Probe:               PROBE10
    Probe status:        enabled
    Latency:             401 ms
    Jitter:              0 ms
    Loss Rate:           - %
    Consecutive Loss:    -
    Combined metric:     3.5/10
  Member:                20
    Next-hop:            172.16.20.1
    Probe:               PROBE20
    Probe status:        enabled
    Latency:             400 ms
    Jitter:              0 ms
    Loss Rate:           - %
    Consecutive Loss:    -
    Combined metric:     2.1/10
```

```
Last Change:
  Current Nexthop:      172.16.10.1
  Previous Nexthop:     -
  Change Time:          22 Nov 2017 13:57:48
  Causes:               Rx probe 'PROBE10', latency (401>300) ms
  Decision:             only available link
Change Count:           1
```

Figure 20: Output parameters for the show pbr rules command

Parameters Explained

PARAMETER	MEANING
Total number of configured PBR-rules	The number of PBR rules currently configured. This includes both conventional PBR policy-routes and linkmon ip policy-routes, regardless of whether the rules are valid or not.
PBR-Rule	The PBR rule ID which the following statistics and configuration is associated with.
Active	Whether the rule is active or not.
Match	The application the PBR policy-route is configured to match on.
From	The source entity the PBR policy-route is configured to match on.
To	The destination entity the PBR policy-route is configured to match on.
Profile	The name of the linkmon profile associated with this linkmon PBR policy-route.
bad above, good above	The configured threshold for this specific rule. There are fields for latency, jitter, and packet loss. If this field has a value of “-”, then the threshold has not been configured.
Consecutive success good when, Consecutive loss bad when, Consecutive loss unreachable when	The configured threshold for consecutive probe loss. If this field has a value of “-”, then the threshold has not been configured.
preference	The preferred metric, to be used in the case of a tie.
Group	The name of the linkmon group associated with this linkmon PBR policy-route.
Member	The ID of the linkmon member associated with this linkmon group.
Next-hop	The configured destination of the linkmon member.
Probe	The linkmon probe associated with the linkmon group member.
Probe status	The status of the probe, enabled or disabled.
Latency	The latency of the probe associated with the linkmon member.
Jitter	The jitter of the probe associated with the linkmon member.

PARAMETER	MEANING
Loss rate	The packet loss of the probe associated with the linkmon member.
Consecutive Loss	The number of consecutive probes lost.
Combined metric	A metric combining latency, jitter, and packet loss. The combined metric is used for tie-breaking. It produces a score out of 10 for the quality of each link, where a score of 0 represents the best possible link and a score of 10 represents the worst.
Current Nexthop	The chosen nexthop for traffic matching the linkmon pbr policy-route.
Previous Nexthop	The previously chosen nexthop prior to failover. If a failover hasn't occurred on this setup, there is no previous nexthop. This is indicated by “-”.
Change Time	The time at which the current nexthop was chosen. This will change if a failover occurs, or at boot.
Causes	The event that caused the last failover.
Decision	The reason why the current nexthop was chosen.
Change Count	The number of times the chosen nexthop has changed. This counter will increment any time a link failover occurs.

show pbr rules brief

Command

```
show pbr rules brief
```

This command shows a summary of all PBR rules, and indicates, by the presence or absence of the nexthop field, which nexthop to route to.

Figure 21: Example output for the show pbr rules brief command

```
awplus#show pbr rules brief
Policy based routing is enabled
Route table usage: 2/62
* - No route table available for the rule - see "show ip pbr route"
Rule Match            From            To              Valid Nexthop
-------------------------------------------------------------------------
 1   any              LAN.VLAN1       any             Yes   -
 10  any              LAN.VLAN1       any             Yes   -
```

Parameters Explained

Figure 22: Output parameters for the show pbr rules brief command

PARAMETER	MEANING
Rule	The PBR rule ID which the following statistics and configuration is associated with.
Match	The application the PBR policy-route is configured to match on.
From	The source entity the PBR policy-route is configured to match on.
To	The destination entity the PBR policy-route is configured to match on.
Valid	Whether the rule is valid or not.
Nexthop	The configured destination of the linkmon member.

show running-config policy-based-routing

Command

```
show running-config policy-based-routing
```

This command is used to show the running system status for policy-based routing on router platforms.

Figure 23: Example output for the show running-config policy-based-routing command

```
awplus#show running-config policy-based-routing
policy-based-routing
 policy-based-routing enable
 ip policy-route 10 from LAN linkmon-group GROUP1 linkmon-profile PROFILE1
 ip policy-route 20 from LAN nexthop 192.168.1.1
```

show ip pbr route

Command

```
show ip pbr route [<1-500>]
```

This command is used to display the installed IPv4 routes for policy-based routing.

Figure 24: Example output for the show ip pbr route command

```
awplus#show ip pbr route
Route table: main
    10.33.11.0/24 via 10.37.236.65, eth1
    10.37.236.64/27 is directly connected, eth1
    172.31.0.0/17 is directly connected, vlan4092
    192.168.1.0/24 is directly connected, vlan2

Route table: policy-route 10

Route table: policy-route 20
    default via 10.37.236.65, ppp0
```

If you do not specify a policy routeID, the output starts by listing the ordinary static and dynamic routes in the route table called “main”. Then it lists the routes for each policy route.

For each route, the output lists the route’s next-hop IP address and/or the next-hop interface.

show linkmon probe

Command

```
show linkmon probe
```

This command is used to display output for one or all link monitoring probes.

Figure 25: Example output for the show linkmon probe command

```
awplus#show linkmon probe
Probe Name : Head-Office-VPN1
  Status : enabled
  Type : ICMP
  IP version : IPv4
  Destination : 198.51.100.1
  Egress Int : -
  Source : -
  DSCP : -
  Packet Size : -
  Interval : -
  Sample Size :
Latest Metrics
  Latency : 1001ms
  Jitter : 0ms
  Packet Loss : 0.0%
Probe Details
  Probes Sent : 3154
  Last Probe Sent : 23 Mar 2018 03:36:00
  Last Probe Received : 23 Mar 2018 03:36:00
```

Figure 26: Output parameters for the show linkmon probe command

PARAMETER	MEANING
Name	The name of the probe.
Status	Whether the probe is enabled or disabled. If it is enabled, then the device will attempt to send probes if the link is up. If it is disabled, then no probes will be sent.
IP version	The IP version being used, IPv4 or IPv6.
Destination	The destination IP address that the probes are sent to.
Egress Interface	The interface that the probe packets should egress.
Source	The source IP address or interface.
DSCP	The DSCP value to use when sending the packet.
Packet Size	The size of a probe packet.
Interval	The number of milliseconds between sending out each probe.
Sample Size	The number of probe results to use when calculating the latency and jitter metrics.
Latency	The average latency based on the last sample size samples.
Jitter	The average jitter based on the last sample size samples.
Packet Loss	The percentage of packets lost based on the last 100 probes.
Probes Sent	The number of probe packets that have been sent.
Last Probe Sent	The time that the last probe packet was sent.
Last Probe Received	The time that the device last successfully received a probe packet.

show running-config linkmon

Command

```
show running-config linkmon
```

This command is used to show the running system status and configuration details for linkmon probes, profiles and groups.

Figure 27: Example output for the show running-config linkmon command

```
awplus#show running-config linkmon
linkmon probe name PROBE10
 destination 10.37.136.98
 enable
!
linkmon probe name PROBE20
 destination 10.37.136.130
 enable
!
linkmon group GROUP1
 member 10 destination tunnel10 probe PROBE10
 member 20 destination tunnel20 probe PROBE20
!
linkmon profile PROFILE1
 latency bad-above 300
```

C613-22106-00 REV D

North America Headquarters | 19800 North Creek Parkway | Suite 100 | Bothell | WA 98011 | USA | T: +1 800 424 4284 | F: +1 425 481 3895
Asia-Pacific Headquarters | 11 Tai Seng Link | Singapore | 534182 | T: +65 6383 3832 | F: +65 6383 3830
EMEA & CSA Operations | Incheonweg 7 | 1437 EK Rozenburg | The Netherlands | T: +31 20 7950020 | F: +31 20 7950021

alliedtelesis.com