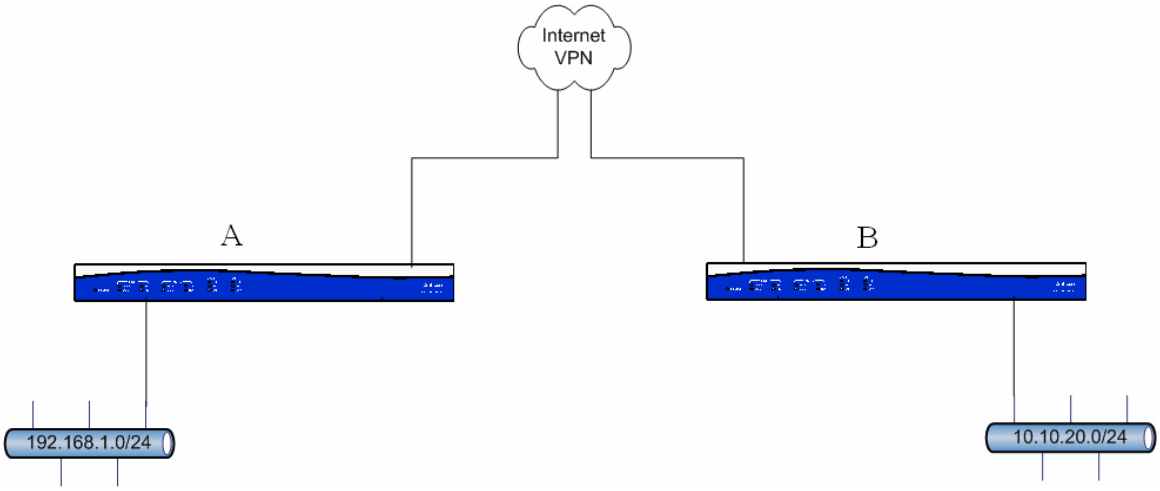
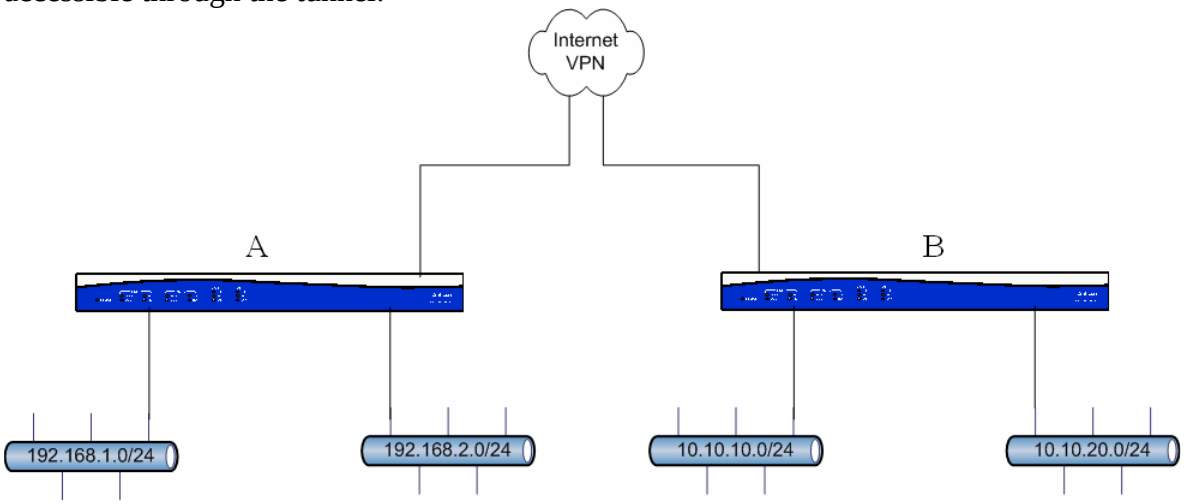
Quick Configuration Guide (QCG)
Configuring a VPN for Multiple Subnets in AOS

Introduction

After creating a VPN, it is often necessary to have access to a new subnet across the VPN. To add a subnet, there must be additional selectors placed in the VPN-selectors. Below is an example network of two routers that each has a single subnet connected.



Above, there is a VPN that is established between 192.168.1.0 /24 and 10.10.20.0 /24. However, as time goes on, there may be a need for an extra subnet at each site to be accessible through the tunnel:



In this scenario, there are now two subnets on both sides that need to be able to reach the remote location. At this point, the selectors must be modified to add an additional subnet

across the link. It is important to note here that by adding an additional subnet to both sides, additional security associations are being added.

Note: Keep in mind the number of security associations that are available on the router. Adding additional subnets to VPN Selectors causes extra processing and extra resources to be used so keep an eye on the number of security associations that are created.

Because the above subnets are discontiguous (10.10.10.0/24 and 10.10.20.0/24), it is necessary to add two statements to allow traffic to pass from Router A to Router B. If these two had been contiguous (s.a. 192.168.0.0/24 and 192.168.1.0/24), it would have been possible to still only use one security association and supernet the two networks (192.168.0.0/23).

Hardware/Software Requirements/Limitations

Software Requirements

The following requirements are needed to be able to establish a multi-network VPN. If the VPN peer is not an AOS device, additional information will be needed:

- Public IP of Statically Addressed Device
- Network IP Addresses and Subnet Masks for all private networks
  - Example: 192.168.1.0 /24 and 10.10.10.0 /24

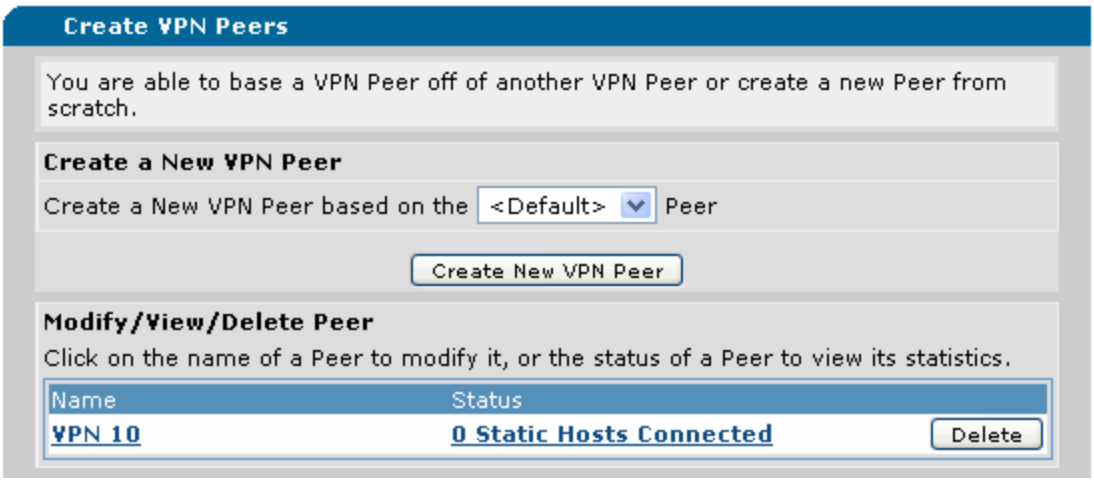
Hardware Requirements

To implement an IPSec VPN tunnel, the following are needed:

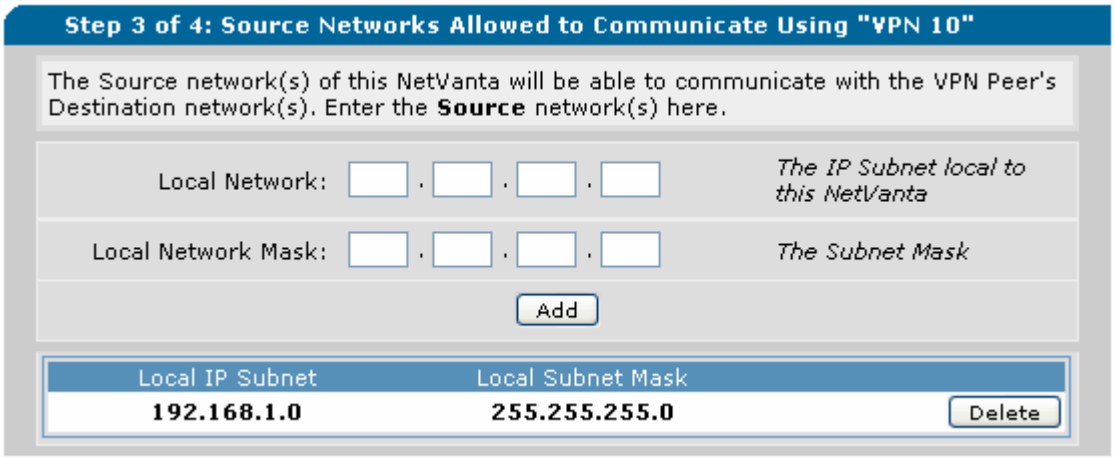
- An Adtran OS Router
  - Example: Netvanta or Total Access
- Enhanced Feature Pack for each Adtran OS Device

Web Interface Configuration

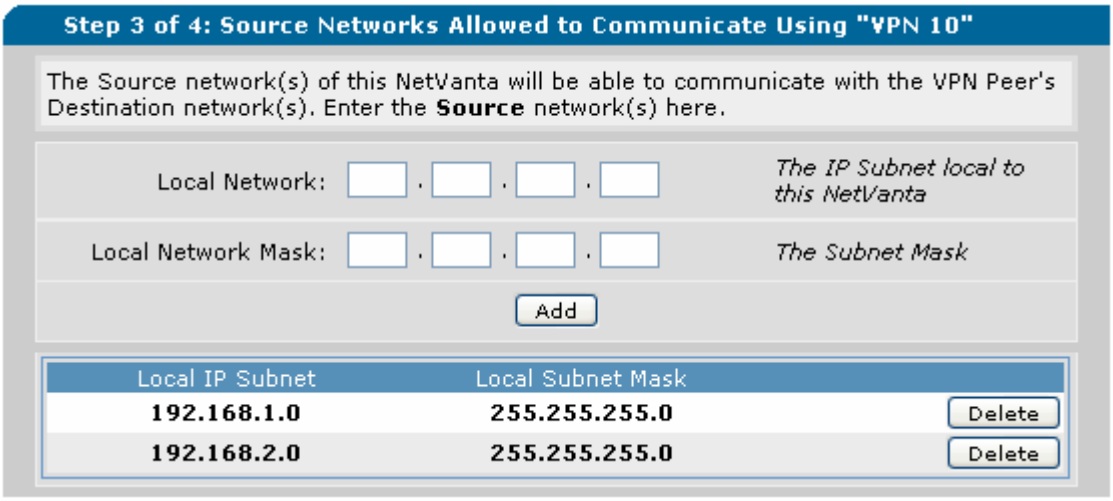
After creating the initial VPN between the two sites, go into the VPN peers to find where additional subnets can be added:



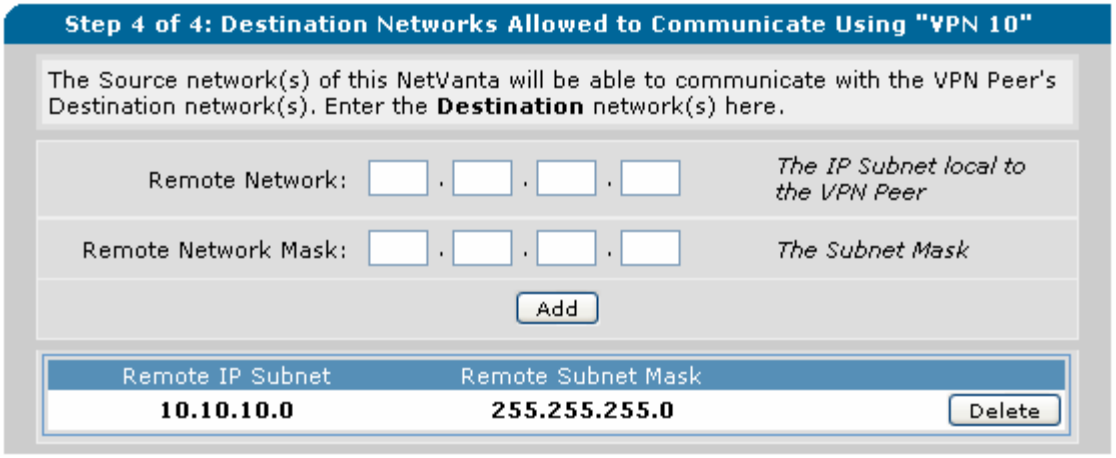
Once this is open, additional subnets (local and remote) can be added to the selectors for the VPN Peer:



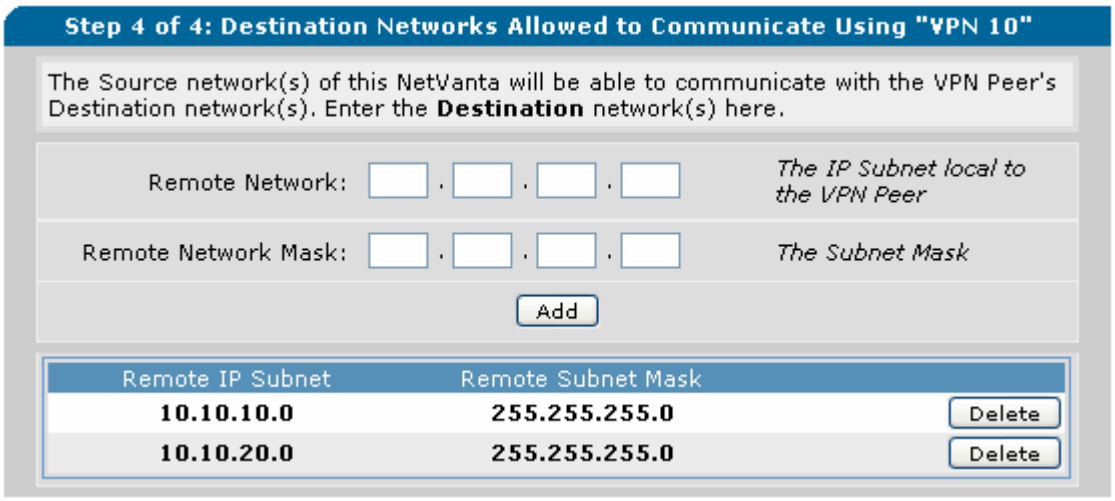
At this point, the 192.168.2.0 subnet needs to be added to the peer so that it can communicate with the 10.10.10.0 subnet. To do this, add the local network of 192.168.2.0 255.255.255.0 under Step 3 in the VPN peers. After typing in the local network and local network mask, click add. Once this is done, both the 192.168.1.0 and 2.0 networks should have access to the 10.10.10.0 subnet:



Finally, to add access for these two networks to the 10.10.20.0 remote network, simply add this remote network to Step 4:



After adding this subnet, all four subnets will have access to one another.



Configuration in CLI

Adding additional subnets only requires one line via the command line for each subnet to get to another remote subnet. Going back to the first example, router A has only one subnet attached. Once the VPN has been created, the access-list for Router A's selectors will look similar to below:

```
ip access-list extended VPN-10-vpnselectors
  permit ip 192.168.1.0 0.0.0.255 10.10.20.0 0.0.0.255
```

The first step to take to get both subnets on A to router B is to add access to the 10.10.10.0 /24 subnet for the 192.168.1.0 /24 subnet:

```
ip access-list extended VPN-10-vpnselectors
  permit ip 192.168.1.0 0.0.0.255 10.10.20.0 0.0.0.255
  permit ip 192.168.1.0 0.0.0.255 10.10.10.0 0.0.0.255
```

At this point, 192.168.1.0 should be able to reach all private subnets on Router B. From here, it is simply a process of adding the source network of 192.168.2.0 to allow access to the remote networks:

```
ip access-list extended VPN-10-vpnselectors
  permit ip 192.168.1.0 0.0.0.255 10.10.20.0 0.0.0.255
  permit ip 192.168.1.0 0.0.0.255 10.10.10.0 0.0.0.255
  permit ip 192.168.2.0 0.0.0.255 10.10.20.0 0.0.0.255
  permit ip 192.168.2.0 0.0.0.255 10.10.10.0 0.0.0.255
```

Once done on router A, the settings can be reversed on router B so that both sides have access to the remote subnets:

```
ip access-list extended VPN-10-vpnselectors
```

permit ip 10.10.20.0 0.0.0.255 192.168.1.0 0.0.0.255
permit ip 10.10.10.0 0.0.0.255 192.168.1.0 0.0.0.255
permit ip 10.10.20.0 0.0.0.255 192.168.2.0 0.0.0.255
permit ip 10.10.10.0 0.0.0.255 192.168.2.0 0.0.0.255

After doing this, security associations should exist allowing access across both local networks to the other remote networks.

Supernetting

Going back to the example above, lets say that instead of 192.168.1.0 /24 and 192.168.2.0 /24, we have 192.168.0.0 /24 and 192.168.1.0 /24. Now it is possible to add a single selector limiting the number of security associations needed.

ip access-list extended VPN-10-vpnselectors
permit ip 192.168.0.0 0.0.1.255 10.10.10.0 0.0.0.255
permit ip 192.168.0.0 0.0.1.255 10.10.20.0 0.0.0.255

In the same light, if subnets 10.10.0.0 /24 and 10.10.1.0 /24 on Router B's side are used, there is only one IPSEC security association in each direction instead of the 4 in the aforementioned section:

ip access-list extended VPN-10-vpnselectors
permit ip 192.168.0.0 0.0.1.255 10.10.0.0 0.0.1.255

Troubleshooting

After setting up the configurations, this can be tested by using the ping utility on the router. Since the tunnel is going to use the same IKE security association for the first tunnel, there shouldn't be any problem with the negotiation. To verify that the VPN selectors match, it still may be necessary to look at the output from **debug crypto ike**. To test this, ping from Router A to Router B and source an internal interface:

ping 10.10.20.1 source 192.168.2.1

If the pings are successful, traffic should pass between the two subnets. If for some reason traffic doesn't pass, use the diagnostic command: **show crypto ipsec sa** from enable mode to verify that the tunnel is up. Below is a sample output from a single security association. From here, it should be possible to verify that packets are being transmitted through the tunnel.

Switch#show crypto ipsec sa
Using 2 SAs out of 2000
IPSec Security Associations:

Peer IP Address: 10.19.243.13

Remote ID: 10.19.243.24
Crypto Map: VPN 10
Direction: Inbound
Encapsulation: ESP
SPI: 0x9010AE7D (2417012349)
RX Bytes: 512
Selectors: Src:192.168.0.0/255.255.254.0 Port:ANY Proto:ALL IP
Dst:10.10.0.0/255.255.254.0 Port:ANY Proto:ALL IP
Hard Lifetime: 28800
Soft Lifetime: 0
Out-of-Sequence Errors: 0

Peer IP Address: 10.19.243.24
Remote ID: 10.19.243.24
Crypto Map: VPN 10
Direction: Outbound
Encapsulation: ESP
SPI: 0xE688CD89 (3867725193)
TX Bytes: 512
Selectors: Src:10.10.0.0/255.255.254.0 Port:ANY Proto:ALL IP
Dst:192.168.0.0/255.255.254.0 Port:ANY Proto:ALL IP
Hard Lifetime: 28800
Soft Lifetime: 28710