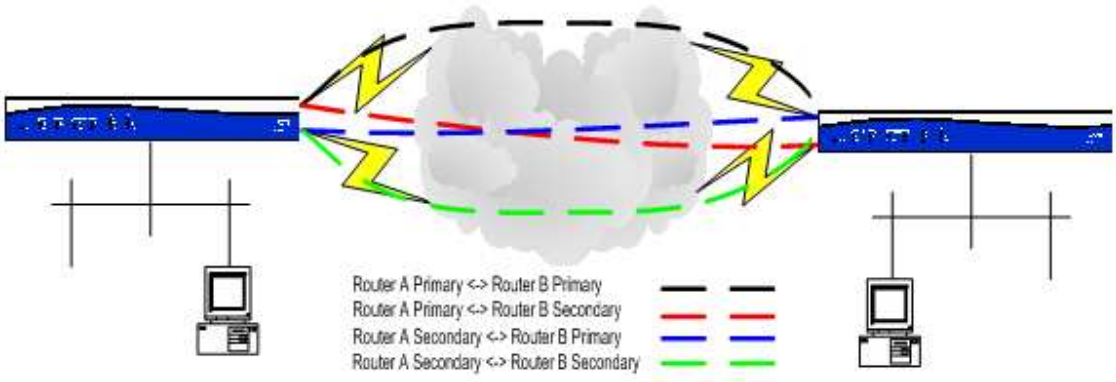
March 2010

Common Application Guide (CAG)
Configuring Redundant VPN Tunnel Fail-Over in AOS

Configuring Redundant VPN Tunnel Fail-Over in AOS



Introduction

Virtual Private Network (VPN) tunnels provide for simulated point-to-point connectivity between two sites through the internet. To minimize downtime, many businesses are moving to multiple Wide-Area Network (WAN) connections. The question then becomes how to make the VPN tunnel fail-over as well. This document will provide the necessary information to allow for properly-constructed and error-free VPN fail-over.

Article Pre-Requisites

This article assumes that the following functions have been configured and understood, some of which will be covered again for clarification on specific issues:

- Multiple internet connections
- Routing fail-over
- Firewall Network Address Translation (NAT) fail-over, if applicable
- Non-fail-over Primary to Primary connection VPN tunnel configuration

Hardware/Software Requirements

VPN tunnel fail-over requires that the following criteria to be met:

- Hardware must support Network Monitor, which includes the 1335, 3000 series (except for the 3200/3205), 4000 series, and 5000 series.
- Device needs to run AOS 15.1 or higher

VPN Tunnel Fail-Over Notes

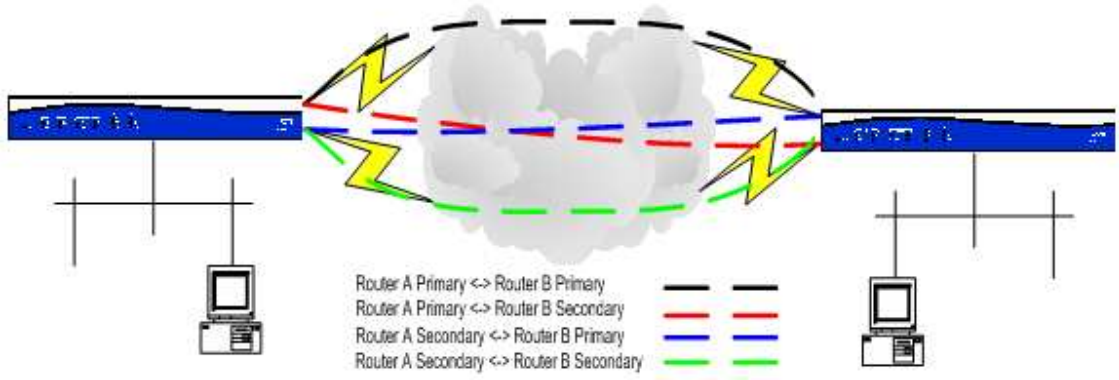
There are a few notes on some of the requirements and caveats to using VPN tunnel fail-over which are listed here:

1) Main mode is the preferred VPN solution because it provides the most protection from attacks and the most security to keep your data away from anyone in the middle of the connection.
    a. Main mode requires that each public interface on the Netvanta have a true public IP, not a private IP with a 1:1 NAT from a public IP. This is due to the limitations in specifying the local ID. If any interface is behind a NAT, Aggressive Mode will need to be used, which is also covered at the end of this article.
2) This configuration is accomplished entirely from the Command Line Interface (CLI). By configuring VPN tunnel fail-over in this manner, the tunnels will no longer appear correctly in the Graphical User Interface (GUI), and should no longer be re-configured using the GUI.

Example Overview

In the example that is presented in this article, two sites, each having two internet connections, want to have a VPN tunnel connection between them, failing over for each possible scenario:

- Site 1 WAN 1 -> Site 2 WAN 1
- Site 1 WAN 2 -> Site 2 WAN 1
- Site 1 WAN 1 -> Site 2 WAN 2
- Site 1 WAN 2 -> Site 2 WAN 2



This is made possible when the following are performed:

- Redundant VPN Tunnel Configuration
- Link State Detection

- Dynamic Removal of a VPN Tunnel
- VPN Keep-Alive

Redundant VPN Tunnel Configuration

Because part of a VPN tunnel's configuration involves specifying a peer IP address or hostname, redundant VPN tunnel configurations must be made for each of a peer's WAN connection. In this case, since each VPN peer has two (2) WAN connections, two (2) VPN tunnel configurations must be programmed.

In many cases, a tunnel from the primary WAN connection to the peer's primary WAN connection can be copied almost one-for-one. A sample configuration of the primary-to-primary tunnel is shown here:

```
ip crypto
!
crypto ike policy 10
  initiate main
  respond main
  peer <Peer Primary WAN IP Address>
  attribute 1
    encryption 3des
    hash md5
    authentication pre-share
!
crypto ike remote-id address <Peer Primary WAN IP Address> preshared-
key <Pre-Shared Key> ike-policy 10 crypto map VPN 10 no-mode-config no-
xauth
!
crypto ipsec transform-set esp-3des-esp-md5-hmac esp-3des esp-md5-hmac
  mode tunnel
!
crypto map VPN 10 ipsec-ike
  match address VPN-10-vpn-selectors
  set peer <Peer Primary WAN IP>
  set transform-set esp-3des-esp-md5-hmac
  ike-policy 10
!
ip access-list extended VPN-10-vpn-selectors
  permit ip <Local Subnet> <Mask> <Peer Subnet> <Peer Mask>
```

To change the configuration to allow redundant connections, there are several changes that need to be made:

- Sequence Numbers
- Peer Statements
- Remote ID(s)
- Crypto Fast-Failover

A sequence number is simply that - a number that indicates the sequence that a configuration should be processed in. They are processed from lowest to highest. To make the configuration easier to read, the best practice is just to add one to the previous sequence number. In this case, that would mean changing each ‘10’ to an ‘11’.

The peer statements indicate the remote party’s IP address, or hostname that resolves to an IP address. Since the peer will be connecting from a different IP address when it is using the backup WAN link, the peer IP will need to be changed to reflect that.

The remote ID(s) that are sent in Main mode must match the peer IP address. Knowing that, a second remote ID statement must be added so the peer’s backup WAN link can be authenticated.

The crypto fast-failover command is very similar to the firewall fast-nat-failover command. It will dynamically tear down all tunnels when a change in the default route is made, which would happen when the track for the primary WAN connection pulls the primary route out of the route table.

One point that needs to be noted is that the Local-ID is not being specified here. This is because this configuration is independent of local WAN connection changes, and only dependant on the peer’s WAN connection changes. The Local-ID(s) being used here is the IP address of the outgoing interface, which is determined by the local route table.

A sample configuration of redundant VPN tunnels is shown here:

```
ip crypto
ip crypto fast-failover
!
crypto ike policy 10
  initiate main
  respond main
  peer <Peer Primary WAN IP Address>
  attribute 1
    encryption 3des
    hash md5
    authentication pre-share
!
crypto ike policy 11
  initiate main
  respond main
  peer <Peer Backup WAN IP Address>
  attribute 1
    encryption 3des
    hash md5
    authentication pre-share
!
crypto ike remote-id address <Peer Primary WAN IP Address> preshared-
key <Pre-Shared Key> ike-policy 10 crypto map VPN 10 no-mode-config no-
xauth
!
crypto ike remote-id address <Peer Backup WAN IP Address> preshared-key
```

```
<Pre-Shared Key> ike-policy 11 crypto map VPN 11 no-mode-config no-
xauth
!
crypto ipsec transform-set esp-3des-esp-md5-hmac esp-3des esp-md5-hmac
  mode tunnel
!
crypto map VPN 10 ipsec-ike
  match address VPN-10-vpn-selectors
  set peer <Peer Primary WAN IP>
  set transform-set esp-3des-esp-md5-hmac
  ike-policy 10
!
crypto map VPN 11 ipsec-ike
  match address VPN-10-vpn-selectors
  set peer <Peer Backup WAN IP>
  set transform-set esp-3des-esp-md5-hmac
  ike-policy 11
!
ip access-list extended VPN-10-vpn-selectors
  permit ip <Local Subnet> <Mask> <Peer Subnet> <Peer Mask>
```

Link State Detection

Each site needs to have knowledge of when the other site's primary link has failed. This is accomplished with network monitor. Network monitor allows the router to 'probe' another device through the use of a ping, TCP connection, or HTTP RAW. In this case, ping probes will be used. The probe is then monitored by a 'track' that can be applied to various functions of the configuration to dynamically remove parts of it from the router's use when the track is in a failed state.

Now the question becomes what needs to be probed. The answer is the peer's primary WAN connection's IP address. A source address will not be specified here because the probe will need to adopt the source IP address of the outgoing interface, since the probe will need to continue to ping the peer's primary WAN even if its own primary connection is down. A sample configuration is shown here:

```
probe VPNPeerWAN1 icmp-echo
  destination <Peer Primary WAN IP Address>
  period 3
  tolerance consecutive fail 3 pass 3
  no shutdown
!
track VPNPeerWAN1
  test if probe VPNPeerWAN1
  no shutdown
```

In this configuration, the probe will fail after three consecutive probes have failed, creating a nine (9) second delay before a failure is detected. This time is user configurable and is based on the needs of each site and what is an acceptable failed state detection time. The risk of lowering the interval increases the chance of creating a false-

positive situation, which could force the VPN tunnel to fail-over when it does not need to, and which the other site may reject, causing the entire tunnel to fail. The equation to figure out the link detection time is:

$$Link\_Detection\_Time(\text{sec}) = Period \times Consecutive\_Interval$$

Dynamic Removal of a VPN Tunnel

The track associated with the probe that was just created can be used to dynamically remove portions of a configuration. In this case, it will be used to remove a particular sequence from a crypto map. A sample configuration is shown here:

```
crypto map VPN 10 ipsec-ike
  match track VPNPeerWAN1
```

By applying the track to the crypto map, it solves the problem of trying to initiate to the peer's primary WAN connection when it is down. Since both sequence 10 & 11 reference the same set of VPN selectors, sequence 11 would never go into effect if sequence 10 was still in the configuration. The IKE policies follow the lead of the associated crypto map(s), and thus there is no need to apply a track to that in this case.

Ensuring Only One VPN Tunnel is Active

This setting is optional, but recommended to ensure that it will not be possible for both tunnels to be UP simultaneously. The same probe that is used to track the primary peer address will be referenced in the inverse by a different track, and that track will be applied to the secondary crypto map. This configuration will ensure that one of the two crypto maps, and their associated tunnels, will always be down. A sample configuration is shown here:

```
track NotVPNPeerWAN1
  test if not probe VPNPeerWAN1
  no shutdown
!
crypto map VPN 11 ipsec-ike
  match track NotVPNPeerWAN1
```

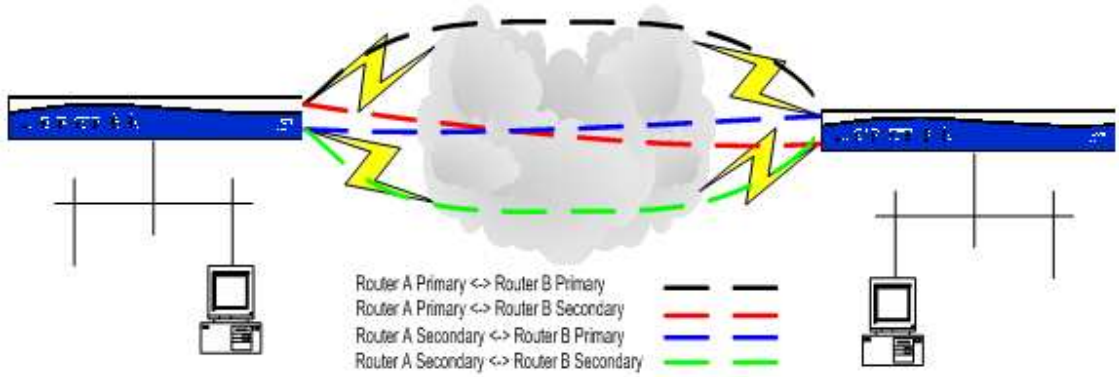
VPN Keep-Alive

This setting is optional, but necessary if having any downtime is a concern. Whenever a tunnel is forced to be brought down, it will not re-negotiate until it receives new 'interesting' traffic. By creating a probe that matches the selector statement, 'interesting' traffic is always created at specified intervals. A sample configuration is shown here:

```
probe VPN-KeepAlive icmp-echo
  destination <Peer Router's LAN IP Address>
  source-address <Router's LAN IP Address>
  period 10
  no shutdown
```

In this case, the keep-alive is sent every 10 seconds. The interval is user-definable and should be set according to your site's needs.

Example Configuration




```
ip local policy route-map LOCAL
!
ip firewall
ip firewall fast-nat-failover
!
probe VPN-KeepAlive icmp-echo
  destination <Peer Router's LAN IP Address>
  source-address <Router's LAN IP Address>
  period 10
  no shutdown
!
probe WAN1 icmp-echo
  destination <WAN 1 Gateway IP>
  source-address <WAN 1 IP Address>
  period 3
  tolerance consecutive fail 3 pass 3
  no shutdown
!
track WAN1
  test if probe WAN1
  no shutdown
!
probe WAN2 icmp-echo
  destination <WAN 2 Gateway IP>
  source-address <WAN 2 IP Address>
```

```
  period 3
  tolerance consecutive fail 3 pass 3
  no shutdown
!
track WAN2
  test if probe WAN2
  no shutdown
!
probe VPNPeerWAN1 icmp-echo
  destination <Peer Primary WAN IP Address>
  period 3
  tolerance consecutive fail 3 pass 3
  no shutdown
!
track VPNPeerWAN1
  test if probe VPNPeerWAN1
  no shutdown
!
track NotVPNPeerWAN1
  test if not probe VPNPeerWAN1
  no shutdown
!
ip crypto
ip crypto fast-failover
!
crypto ike policy 10
  initiate main
  respond main
  peer <Peer Primary WAN IP Address>
  attribute 1
    encryption 3des
    hash md5
    authentication pre-share
!
crypto ike policy 11
  initiate main
  respond main
  peer <Peer Backup WAN IP Address>
  attribute 1
    encryption 3des
    hash md5
    authentication pre-share
!
crypto ike remote-id address <Peer Primary WAN IP Address> preshared-
key <Pre-Shared Key> ike-policy 10 crypto map VPN 10 no-mode-config no-
xauth
!
crypto ike remote-id address <Peer Backup WAN IP Address> preshared-key
<Pre-Shared Key> ike-policy 11 crypto map VPN 11 no-mode-config no-
xauth
!
crypto ipsec transform-set esp-3des-esp-md5-hmac esp-3des esp-md5-hmac
  mode tunnel
!
crypto map VPN 10 ipsec-ike
  match track VPNPeerWAN1
  match address VPN-10-vpn-selectors
```

```
  set peer <Peer Primary WAN IP>
  set transform-set esp-3des-esp-md5-hmac
  ike-policy 10
!
crypto map VPN 11 ipsec-ike
  match track NotVPNPeerWAN1
  match address VPN-10-vpn-selectors
  set peer <Peer Backup WAN IP>
  set transform-set esp-3des-esp-md5-hmac
  ike-policy 11
!
interface eth 0/1
  description Primary WAN Connection
  ip address  <WAN 1 IP> <WAN 1 Subnet Mask>
  access-policy Public
  crypto map VPN
  no shutdown
!
interface eth 0/2
  description Secondary WAN Connection
  ip address  <WAN 2 IP> <WAN 2 Subnet Mask>
  access-policy Public-Backup
  crypto map VPN
  no shutdown
!
interface vlan 1
  ip address  <Router's LAN IP Address> <Subnet Mask>
  access-policy Private
  no shutdown
!
route-map LOCAL permit 10
  match ip address WAN1
  set ip next-hop <WAN 1 Gateway IP>
  set interface null 0
!
route-map LOCAL permit 20
  match ip address WAN2
  set ip next-hop <WAN 2 Gateway IP>
  set interface null 0
!
ip access-list extended WAN1
  permit icmp host <WAN 1 IP Address> host <WAN 1 Gateway IP>
!
ip access-list extended WAN2
  permit icmp host <WAN 2 IP Address> host <WAN 2 Gateway IP>
!
ip access-list standard NAT1
  permit any
!
ip access-list standard NAT2
  permit any
!
ip access-list extended AdminAccess
  permit tcp any  any eq www   log
  permit tcp any  any eq telnet   log
  permit udp any  any eq snmp    log
  permit tcp any  any eq https   log
```

```
  permit tcp any  any eq ssh   log
  permit tcp any  any eq ftp   log
  permit icmp any  any  echo   log
!
ip access-list extended self
  remark Traffic to Netvanta
  permit ip any  any
!
ip access-list extended VPN-10-vpn-selectors
  permit ip <Local Subnet> <Mask> <Peer Subnet> <Peer Mask>
!
ip policy-class Private
  allow list VPN-10-vpn-selectors stateless
  allow list self self
  nat source list NAT1 interface eth 0/1 overload policy Public
  nat source list NAT2 interface eth 0/2 overload policy Public-Backup
!
ip policy-class Public
  allow reverse list VPN-10-vpn-selectors stateless
  allow list AdminAccess self
!
ip policy-class Public-Backup
  allow reverse list VPN-10-vpn-selectors stateless
  allow list AdminAccess self
!
ip route 0.0.0.0 0.0.0.0 <WAN 1 Gateway IP> track WAN1
ip route 0.0.0.0 0.0.0.0 <WAN 2 Gateway IP> 10 track WAN2
```

Aggressive Mode Configuration

If main mode is not an option because one of the interfaces on your router is behind a NAT, this will show you how to convert the router configuration to aggressive mode. In this scenario, we will still be specifying the peer IP address, so we are not as susceptible to attack, but aggressive mode tunnels, due their nature, are not as secure as a main mode tunnel and could make it easier for a third party to look at the data going across the tunnel.

The following changes will need to be made:

- IKE policies will need to be set to use aggressive mode
- Specify the ID that the IKE policy will send
- Change the remote ID statements to reflect those ID changes

```
crypto ike policy 10
  initiate aggressive
  respond aggressive
  local-id fqdn <Primary FQDN>
  peer <Peer Primary WAN IP Address>
  attribute 1
    encryption 3des
    hash md5
```

```
    authentication pre-share
!
crypto ike policy 11
  initiate aggressive
  respond aggressive
  local-id fqdn <Secondary FQDN>
  peer <Peer Secondary WAN IP Address>
  attribute 1
    encryption 3des
    hash md5
    authentication pre-share
!
crypto ike remote-id fqdn <Peer Primary FQDN> preshared-key <Pre-Shared
Key> ike-policy 10 crypto map VPN 10 no-mode-config no-xauth
!
crypto ike remote-id fqdn <Peer Secondary FQDN> preshared-key <Pre-
Shared Key> ike-policy 11 crypto map VPN 11 no-mode-config no-xauth
```

Notice that not much has changed. The only real difference is that Fully-Qualified Domain Names (FQDNs) are now used instead of IP addresses. Main mode does not allow for any ID other than IP address and the ID has to match the IP address the tunnel is coming from. Aggressive mode has no such rule and allows for any type of ID.

NOTE: *The FQDN does not have to be real. The router is simply sending a string of characters that have to match what the other side is expecting. There is no third party involved, nor is there a DNS lookup performed.*