

Meet Me Zone Design

CONTENTS

Overview.....	4
Background.....	4
Terminology.....	4
Intent	4
Salient Features.....	5
Firewalls.....	5
The Pit.....	5
Site-to-Site IPsec VPN Tunnels.....	6
Guest Network	6
Remote Access VPN.....	7
WHI.....	7
Routing Design	7
External & Internal	7
Exterior Gateway Protocol.....	7
Interior Gateway Protocol.....	7
Edge Routers	7
Conversing with the External World	7
Conversing with the Internal World	8
Guest Network	8
Core Routers	8
Border VRF.....	9
Native VRF	10
Gateway-of-Last-Resort.....	11
BGP.....	12
EIGRP	13
High-Availability	18
Load Balanced vs Active/Standby	19
Border VRF.....	19
Native VRF	19
Validation.....	20
Understanding the IPsec SPA Modules	20

Configuring VPNs in Crypto Connect Mode.....	20
IPsec LAN-to-LAN Tunnel Between a Catalyst 6500 with the VPN Service Module and a Cisco IOS Router Configuration Example	22
IPsec Stateful Failover (VPN High Availability) Feature Module.....	22
Configuring Duplicate Hardware and IPsec Failover Using the IPsec VPN SPA	23
Troubleshooting the IPsec VPN SPA	23
Explore Tunnels	23
Examine Hardware.....	23
Basic Health	25
Crypto Status in Detail.....	26
mmz-a-rtr	26
mmz-b-rtr	28
Stateful Synchronization Protocol	28
Crypto and SSP Syslog Messages.....	32
Normal	32
Failover	32
Failback.....	32
Explore Neighbors	34
Native VRF	34
Border VRF.....	36
Explore Routing	36
HSRP.....	36
Virtualized Route Table.....	38
Examining the Route Tables.....	39
Walk the Route Tables.....	40
mmz-x-rtr Native VRF.....	40
mmz-x-rtr Border VRF	43
internap-a-rtr	48
internap-b-rtr	49
gigapop-a-rtr	50
gigapop-b-rtr	52
Explore VLANs	53
Overview.....	53
Native VRF	56

Border VRF.....	57
Hacks.....	58
mac-address.....	58
Disable CDPv2.....	59
Guest Network	59
Annoying Log Messages.....	59
Walk the Config Files	60
mmz-a-rtr	60
gigapop-a-rtr	86
internap-a-rtr	96

OVERVIEW

This document describes how and why we have designed the MMZ.

BACKGROUND

Terminology

Traditionally, the industry called a transport-only network connecting an enterprise to its partners and its service providers a De-Militarized Zone (DMZ), a term borrowed from the Korean war. However, in recent years, this term has come to mean other things, typically a place where externally accessible servers sit. In an effort (possibly unsuccessful) to more precisely specify what we intend in this section of our network, we changed the name from DMZ to *Meet Me Zone* (MMZ). And we call the place where externally accessible servers sit *The Pit*.

The 'core' routers in the MMZ are named *mmz-x-rtr*. The 'edge' routers are *internap-x-rtr*, *gigapop-x-rtr*, and *manwe*. The remote access servers are *charon-x-vpn*; they provide a path for https and IPsec based remote access telecommuters to enter the Hutch network (bypassing the Hutch firewall). And the firewalls themselves are named *ice-x-fw*, *ga-x-fw*, and *cf-ptl/md-ptl*¹ and provide filtered access to their respective networks (WIDGETS, GADGETS, and WIDGETS Guest).

manwe is a router owned and managed by the Women's Health Initiative (WHI), the 'Coordinating Center' chunk of which is a Hutch-based project which is contractually bound to manage its own data network; we meet in the MMZ.

Intent

We intend the MMZ to be the logical location where the Hutch and the GADGETS exchange traffic with each other, with our service providers (Internet service providers and regional transit providers), with partners, and with independent entities hosted at our two organizations.

We intend the MMZ to be shared between the Hutch and the GADGETS, leveraging our close relationship to reduce costs.

We intend the MMZ to be highly available, specifically, to be able to suffer the failure of any single device without disrupting end-user service. To this end, we invest significant effort into a design employing highly-available techniques, hardening its services against the various device and software failure modes. And, we have split the MMZ gear across three physical locations, physically near the three points at which service providers deliver glass to the campus, in order to harden its services against physical layer disruption.

¹ These devices also function as NAT gateways, Guest Authentication Portals, and DHCP servers.

As part of this highly-available strategy, we compartmentalize MMZ functions, strictly defining the interface between elements, in an effort to make the whole more easily understandable and fixable.

Implications

- This is the place where we terminate site-to-site VPN tunnels. We decrypt incoming traffic inside the MMZ; such traffic must then traverse the Firewalls in order to arrive at its destination. Thus, the site-to-site VPN tunnels provide a method for encrypting traffic traversing public networks; they do not provide an encrypted tunnel through GADGETS or Hutch corporate firewalls.
- We centralize our perimeter access policy in the Firewall rule-sets, not in the site-to-site VPN crypto policies.
- On the inside, firewalls shield corporate networks (*ice-x-fw* for the Hutch, *ga-x-fw* for the GADGETS, *cf-ptl/md-ptl* for the Hutch Guest Wireless network).
- At the edge, the border routers (*internap-x-rtr*, *gigapop-x-rtr*, *manwe*) meet the service providers and independent WIDGETS entities.
- The MMZ provides transport only services; it does not host end-stations.

Salient Features

FIREWALLS

A pair of highly-available firewalls (CheckPoint on Nokia IP560) provide access to WIDGETS's network; a second pair of highly-available firewalls (Cisco PIX 535) provide access to the GADGETS's network.

The firewalls are configured as active/standby pairs. On the GADGETS side, we employ the PIX's stateful failover feature; on the Hutch side, we use a CheckPoint specific feature (established TCP connections) to provide a facsimile of stateful failover.

For WIDGETS's Guest Network, the firewalls are not currently highly-available (work-in-progress) but must be manually engaged.

THE PIT

Dangling off the side of the Hutch firewalls is *The Pit*, a place where some groups place externally accessible servers. Originally, the firewalls were to permit limited inbound connections and no outbound connections. However, over time, we have eroded this stance and at this point, many of the boxes inside *The Pit* can initiate connections not only to the outside world but also internally, into WIDGETS. Essentially, we do not yet understand how to segregate external servers from internal servers.

In some ways, *The Pit* is configured to look like a typical server room – a pair of Ethernet switches connected to a pair of routers (well, stateful firewalls). However, we are struggling to make the transport side of *The Pit* as highly-available as a typical server room. The details of

how the firewalls fail over from active to standby depending on the failures of gear around them (switches like those in *The Pit* and routers like those bracketing the firewalls) have lead us to make trade-offs. At the moment, if the 'a' switch in *The Pit* goes down, *The Pit* is isolated from the rest of the world.

SITE-TO-SITE IPSEC VPN TUNNELS

The IPsec SPA modules in *mmz-x-rtr* function as encryption off-load engines, off-loading the calculation-intensive work of encrypting/decrypting packets from the main CPU on the Supervisor card. Originally, we intended these modules to service site-to-site VPN tunnels, instituted after the creation of the GADGETS, to encrypt sensitive traffic crossing the PNW GigaPOP, i.e. between the GADGETS and its partners at CHRMC and the UW. Today, most of the devices employing these tunnels are located inside the GADGETS's network; however, some of them are located inside the Hutch's network. In addition, some GADGETS service providers (vendors who support their gear installed at the GADGETS) manage their gear installed in GADGETS facilities via site-to-site VPN tunnels. In general, the GADGETS has access to all the tunnels; the Hutch has access to some of the tunnels.

Only one IPsec SPA module is active at a time; we employ HSRP on both the inside (Hutch & GADGETS) and outside (129.34.8.201) facing interfaces to enforce this. We employ stateful failover to minimize service disruption when switching from one IPsec module to the other.

Much of the MMZ is constructed around the specialized requirements of these devices.² In choosing this hardware, we have opted for performance over simplicity.

GUEST NETWORK

The Guest Network captive portals exit in the MMZ. We have borrowed a pair of /29 spaces from our ISP, and the captive portals employ these spaces to NAT their guest clients to public addresses. We do this in an effort to disassociate ourselves with Guest traffic -- although Guest traffic traverses our MMZ and exits to the Internet via circuits which also carry corporate traffic, the sources addresses do not belong to the IP spaces registered to WIDGETS.

Notice that Guest traffic headed toward WIDGETS must traverse not only *mmz-x-rtr* but also the Firewalls, just like any other traffic arriving from the Internet as a whole.

As of this writing, the GADGETS Guest Network employs public IP space registered to the GADGETS, employs *ga-x-fw* as the NAT devices, and cannot access GADGETS resources.

² IMHO, Cisco aimed these modules at the carrier space, a space which uses them in one or two specific deployment scenarios to deliver high volume solutions (high throughput, many tunnels), typically involving dedicated chassis. They do not fit our deployment scenario particularly well -- integrated in the MMZ's core routers -- ergo the use of VRFs to integrate them into the design.

REMOTE ACCESS VPN

On the WIDGETS side, the remote access VPN servers bypass the firewalls, providing their own path from the MMZ to the internal enterprise network.

WHI

WHI meets the rest of the Hutch in the MMZ. They manage 129.34.8.208/28, 129.34.8.224/27, and 129.34.9.128/25, NATing their end-stations into this space. This gives their users 129.34.0.0/16 addresses, which is useful when employing sites which restrict access based on IP space (typically resources rented by the Library). Notice that WHI users must traverse the Firewalls, if they want to reach points inside WIDGETS or the GADGETS.

ROUTING DESIGN

External & Internal

EXTERIOR GATEWAY PROTOCOL

The Edge Routers peer with their service provider counterparts (eBGP), accepting whatever routes they send and forwarding those routes (iBGP) to *mmz-x-rtr*, which function as route reflectors. We purchase Commodity Internet access from the InterNAP; as a result, the InterNAP advertises gateway-of-last-resort to us, which we accept. We purchase only local service from the PNW GigaPOP – in GigaPOP terms, this translates into access to roughly a third of the Commodity Internet. We rely on BGP to pick the best path outbound, and bias our route advertisements to the InterNAP (AS path pre-pend) to make the InterNAP in-bound paths more costly than the PNW GigaPOP paths.

INTERIOR GATEWAY PROTOCOL

Internally, the MMZ boxes (both Edge Router and Core Routers) employ EIGRP to exchange local routes. With WHI, we send only gateway-of-last-resort, and we accept only their three IP subnets. With the Guest Network portals and the Firewalls, we exchange no routes, relying instead on statics.

We use filters to shrink the route tables (effectively to aggregate MMZ-only routes): smaller is better.

Edge Routers

CONVERSING WITH THE EXTERNAL WORLD

The Edge Routers *internap-x-rtr* and *gigapop-x-rtr* speak eBGP with their partners at our service providers. They implement the usual edge security filters, protecting us from IP spoofing, throwing away bogons, and protecting the MMZ interfaces themselves from attacks.

manwe doesn't speak with anyone else internally -- WHI uses static routes internally. The Core Routers (*mmz-x-rtr*) use filters to advertise only gateway-of-last-resort to *manwe* and to accept advertisements only for 129.34.8.208/28, 129.34.8.224/27, and 129.34.9.128/25, the IP space we've allocated to WHI, from *manwe*.

The Guest Network firewalls/portals (*cf-ptl* and *md-ptl*) also connect at the Edge; in this sense, we treat the Guest Network as an 'external' network, similar to a partner (like WHI). Traffic heading from the Guest Network toward the Hutch must traverse the Border VRF, just like all other Internet traffic.

CONVERSING WITH THE INTERNAL WORLD

Each of the Edge Routers (except for *cf-ptl/md-ptl*) connect to the Core Routers -- specifically, to the Border VRF in the Core Routers -- and exchange routes via EIGRP.

GUEST NETWORK

The firewalls/portals (*cf-ptl* and *md-ptl*) leading to the Guest Network attach here, one plugged into *internap-a-rtr* and the other plugged into *internap-b-rtr*. I am not entirely happy with this choice.

On the one hand, since the IP spaces which these portals control are provided by the InterNAP and must be routed to and from the InterNAP, they are, in some sense, tied rather tightly to our InterNAP service. The physical connectivity here reflects this bond.

On the other hand, it seems gross to me, to glue these devices to the side of *internap-x-rtr*.

An alternate approach would be to plug them into the Border VRF portion of *mmz-x-rtr* and build GRE tunnels across the Border VRF to *internap-a-rtr* and *internap-b-rtr* respectively. Today, I don't see that this would buy us additional functionality, which is why I opted to go for the aesthetically displeasing but less complicated approach of gluing the portals to *internap-x-rtr*.

If, in the future, we see a need to expand the Guest Network (perhaps by adding an GADGETS component), then we might take this approach. Alternatively, we could add more ports to *internap-x-rtr* (I note that *internap-a-rtr* already contains an unused Fast Ethernet port). Or acquire sliver subnets from the GigaPOP and plug the portals into *gigapop-x-rtr*.

Core Routers

From a routing point of view, we have employed VRF-Lite to saw *mmz-x-rtr* in half, into a Border VRF and a Native VRF.³ The Border VRF handles the route reflector function; the

³ 'Native VRF' is a term I first encountered from Brad Jordan of Advanced Technology Partners. Garth Brown of Semaphore seems to understand it. Cisco documentation tends to refer to it as the 'default VRF' ... if they discuss it at all. And I see only a single reference to this concept via a broad Google search ... from a posting at the UW, where Brad worked for some years.

Native VRF contains the IPsec modules and handles connectivity to the corporate networks (meets the Remote Access Servers and the Firewalls). The two VRFs are tied together via two paths (aka *handle-bars*): one carrying normal traffic, the other carrying encrypted traffic. These paths are not interchangeable – if one goes away, then we rely on the IGP to route traffic to the surviving path in the intact router. e.g. if the Normal Traffic handlebar on *mmz-a-rtr* breaks, then we rely on the IGP to push traffic across the *mmz-a-rtr* <==> *mmz-b-rtr* interlink to the Normal Traffic handlebar on *mmz-b-rtr* and from there to the Border VRF.

BORDER VRF

In a BGP sense, the Border VRF owns AS 14954 and the IP super-nets (129.34.0.0/16 and 65.90.32.0/19) which it advertises.

Edge

The Border VRF in *mmz-x-rtr* mediates all conversation with the Edge Routers – the Edge Routers talk to *mmz-x-rtr* via point-to-point connections; they do not exchange routes directly with one another. The Border VRF portion of *mmz-x-rtr* function as the (Layer 3) Core to the MMZ, borrowing a term here from Network Design 101. External traffic (including Guest Network traffic) must traverse the Border VRF within *mmz-x-rtr* and the Native VRF within *mmz-x-rtr* before approaching the corporate Firewalls.

Conversing with the Border VRF

The Border VRF in *mmz-x-rtr* exchange routes (through both IGP and EGP) with each other via VLAN401 (129.34.8.40/31), a VLAN which rides across the EtherChannelled pair of GigE connections tying *mmz-x-rtr* together.

Conversing with the Native VRF

The Border VRF exchanges routes (and traffic) with the Native VRF via the Normal Traffic handle-bars, which, for the Border VRF, terminate in the VLAN308 interfaces (129.34.8.2/31 for *mmz-a-rtr*, 129.34.9.2/31 for *mmz-b-rtr*). On the Native VRF side, these handle-bars are fed by the VLAN307 interfaces.⁴ Filters restrict the advertisements from the Native VRF to 129.34.0.0/16 and 65.90.32.0/19, the two super-routes for which the Native VRF acts as gatekeeper.

The Border VRF also exchanges traffic with the Native VRF via its VLAN 306 (129.34.8.192/28) interfaces – but this path carries only IPsec traffic. On the Border VRF side, these interfaces are Layer 3 VLAN interfaces (129.34.8.194 for *mmz-a-rtr*; 129.34.8.195 for

⁴ I find this confusing – why not have the same VLAN on both sides of this cable? The reason is that Cisco's VRF function virtualizes the IP routing table but not the VLAN address space. So the virtualization isn't perfect – all VRFs draw from the same pool of 4096 VLAN numbers. Each interface must be a Layer 3 interface (in order to exchange routes across the VRF barrier). We could make this a 'raw' interfaces, i.e. just Gigabit Ethernet interfaces with IP addresses. At the moment, I have made them Layer 3 VLAN interfaces, in order to remain consistent with the other routed interfaces (which are also Layer 3 VLAN interfaces). But doing this leads to mis-matched VLANs on either side of this cable. As a result, *mmz-x-rtr* run CDP v1 (which doesn't notice VLAN mismatches) rather than CDP v2 (which does). And I experience this cognitive jarring every time I look at the diagram and notice that the VLAN numbers don't match across the handle-bar interfaces.

mmz-b-rtr). On the Native VRF side, these VLAN interfaces (VLAN 305) are Layer 2 VLAN interfaces. And VLAN 305 rides across the EtherChanneled interconnect, providing a single broadcast domain for the two routers (*mmz-x-rtr*) over which to exchange HSRP Hellos, providing highly-available access to 129.34.8.201: this is the IP address at which our partners aim their IPsec tunnels.

Notice how VLAN 306, 307, and 308 exist in both *mmx-x-rtr*, yet are not shared. For example, the *mmz-a-rtr* VI306 interface and the *mmz-b-rtr* VI306 interface do not share a broadcast domain. I toyed with the idea of using unique vlan numbers, but ended up with this approach, believing it to be easier to understand (one can more quickly grasp the parallelism between the two devices, I believe).

NATIVE VRF

In a functional sense, the Native VRF owns our public IP spaces (129.34.0.0/16 and 65.90.32.0/19), acting as a gateway to the Firewalls which control access to these networks. As a reflection of this role, the Native VRF contains static routes which send traffic for these supernets to *ice-x-fw* and *ga-x-fw*.

As a way to reduce the effect of DoS attacks, we employ static routes to Null0 in the Native VRF to discard in-bound traffic headed to not-yet-defined subnets inside the larger 129.34.0.0/16 and 65.90.32.0/19 spaces, as well as to throw away bogons. Bogons, of course, should have been tossed by the Edge Routers (incoming) or by the Firewalls (outgoing); these Null0 routes function as a second line of defense, in case the primary filters are accidentally removed.

Additionally, the Native VRF contains Null0 routes for our public IP spaces (with a high administrative weight), allowing *mmz-x-rtr* to efficiently discard traffic which is unroutable during major failure events.

Firewalls

The Native VRF sits just in front of the Firewalls, giving them a highly-available Layer 2 subnet across which they can exchange their VRRP (or equivalent) traffic. 129.34.0.0/28 (VLAN 301) delivers this service to *ice-x-fw*; 129.34.0.16/28 (VLAN 302) delivers this service to *ga-x-fw*. *mmz-x-rtr* employ HSRP to present a single next hop address to the Firewalls in each of these subnets; VLANs 301 & 302 ride across the EtherChanneled interlink between *mmz-x-rtr*.

Site-to-Site VPN Tunnels

The Native VRF also contains the IPsec modules. Policy-based route maps on the Firewall-facing interfaces (VLAN301/302) direct traffic destined for protected subnets across the IPsec modules and toward 129.34.8.194 (*mmz-a-rtr*) or 129.34.8.195 (*mmz-b-rtr*). As the traffic traverses the IPsec modules, they encrypt it, and the resulting IPsec frames reach the Border VRF on the VLAN306 interfaces.⁵ Inbound, traffic departs from the Layer 3 VLAN 306

⁵ Notice that while the VLAN address space is unique within a single MMZ router, they are *not* unique between routers – VLAN 306 – 308 appear in *both* *mmz-x-rtr*, where they employ identical functions. Confusingly, they do not share a broadcast domain.

interfaces, crosses VLAN 305, and crosses the IPsec module to reach 129.34.8.201, which is hosted on the 'inside' interface of the IPsec module (Layer 3 VLAN 303 interface). As the frame crosses the 'bump-in-the-wire' IPsec module, it is decrypted and then forwarded using the Native VRF's routing table.

When the IPsec modules build tunnels with their remote partners, they use the 'reverse-route' feature to inject routes to the protected subnets into the Native VRFs route table. In the general case, this step is unnecessary, since the policy-based route maps on the Firewall-facing interfaces handle the task of propagating traffic across the IPsec module appropriately. However, Brad McGaugh of ATP recommended doing this, in order to handle transient situations in which an IPsec module is not functioning in the receiving router but while its partner in the other router is alive and well.⁶

Remote Access

Finally, the Native VRF also hosts the interfaces to the Remote Access Servers, providing an alternate path (bypassing the Firewalls) for telecommuter traffic. This subnet (129.34.0.48/28, VLAN310) rides across the EtherChanneled interlink, providing an HSRP address to which the Remote Access Servers forward traffic.

Conversing with the Native VRF

The Native VRF IGP processes on *mmz-x-rtr* exchange routes with each other across VLAN 400 (129.34.0.40/31), a VLAN which rides across the EtherChannel. A 'distribute-list' on this conversation artificially increases cost, in order to shrink the size of the IGP-derived routing table within the Native VRF.

Conversing with the Border VRF

The Native VRF exchanges routes with the Border VRF across its VLAN 307 interface (129.34.8.2/31). Filters restrict in-bound routes to just 129.34.8.0, 129.34.9.0, and gateway-of-last-resort, in order to shrink the size of the Native VRF's route table. Regrettably, I have not figured out how to filter outbound routes, so the Border VRF contains unnecessary /32 and /28 routes to 129.34.0.0 subnets.

Gateway-of-Last-Resort

The InterNAP advertises gateway-of-last-resort to us via BGP, along with all the other routes they hand us. *internap-x-rtr* propagate this advertisement to *mmz-x-rtr*, where we redistribute it into EIGRP inside the Border VRF. EIGRP in the Border VRF then advertises it to the Native VRF.

⁶ I have not tested the utility of this design choice. --sk

BGP

internap-x-rtr

The BGP process inside *internap-x-rtr* hear about 0.0.0.0 and advertises this route to its Route Reflectors (fortunately, it does **not** re-advertise it back to its peer at the InterNAP).

```
internap-a-rtr#sh ip bgp 0.0.0.0
BGP routing table entry for 0.0.0.0/0, version 2
Paths: (1 available, best #1, table Default-IP-Routing-Table, not advertised
to EBGp peer)
```

```
  Advertised to peer-groups:
    route-reflectors
```

```
  14744
```

```
    64.252.132.150 from 64.252.132.150 (63.251.160.139)
```

```
      Origin IGP, metric 0, localpref 100, valid, external, best
```

```
      Community: no-export
```

```
internap-a-rtr#
```

```
internap-b-rtr#sh ip bgp 0.0.0.0
```

```
BGP routing table entry for 0.0.0.0/0, version 2
```

```
Paths: (1 available, best #1, table Default-IP-Routing-Table, not advertised
to EBGp peer)
```

```
  Advertised to peer-groups:
    route-reflectors
```

```
  14744
```

```
    206.191.144.137 from 206.191.144.137 (206.253.192.12)
```

```
      Origin IGP, metric 0, localpref 100, valid, external, best
```

```
      Community: no-export
```

```
internap-b-rtr#
```

And *internap-x-rtr* insert this route into their routing tables:

```
internap-a-rtr#sh ip ro 0.0.0.0
```

```
Routing entry for 0.0.0.0/0, supernet
```

```
  Known via "bgp 14954", distance 20, metric 0, candidate default path
```

```
  Tag 14744, type external
```

```
  Last update from 64.252.132.150 02:39:04 ago
```

```
  Routing Descriptor Blocks:
```

```
  * 63.251.162.149, from 63.251.162.149, 02:39:04 ago
```

```
    Route metric is 0, traffic share count is 1
```

```
    AS Hops 1
```

```
internap-a-rtr#
```

```
internap-b-rtr#sh ip ro 0.0.0.0
Routing entry for 0.0.0.0/0, supernet
  Known via "bgp 14954", distance 20, metric 0, candidate default path
  Tag 14744, type external
  Last update from 206.191.144.137 02:33:36 ago
  Routing Descriptor Blocks:
  * 206.191.144.137, from 206.191.144.137, 02:33:36 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
```

```
internap-b-rtr#
```

mmz-x-rtr

In turn, the BGP processes in *mmz-x-rtr* hear about 0.0.0.0 from *internap-x-rtr*. Notice the additional CLI syntax imposed by running BGP inside a VRF. Notice also the *mmz-x-rtr* do **not** re-advertise 0.0.0.0 to any peers -- they are big, bad Route Reflectors, after all, and are above such behavior.

```
mmz-a-rtr#sh ip bgp vpv4 vrf Border 0.0.0.0
BGP routing table entry for 65535:1:0.0.0.0/0, version 190653
Paths: (2 available, best #1, table Border, not advertised to EBGp peer)
  Not advertised to any peer
```

```
14744
  64.252.132.150(metric 63242) from 129.34.8.105 (129.34.8.105)
    Origin IGP, metric 0, localpref 100, valid, internal, best
    Community: no-export
14744
  206.191.144.137 (metric 63242) from 129.34.8.108 (129.34.8.108)
    Origin IGP, metric 0, localpref 100, valid, internal
    Community: no-export
```

```
mmz-a-rtr#
```

```
mmz-b-rtr#sh ip bgp vpv4 vrf Border 0.0.0.0
BGP routing table entry for 65535:1:0.0.0.0/0, version 259779
Paths: (2 available, best #2, table Border, not advertised to EBGp peer)
  Not advertised to any peer
```

```
14744
  206.191.144.137 (metric 63242) from 129.34.8.108 (129.34.8.108)
    Origin IGP, metric 0, localpref 100, valid, internal
    Community: no-export
14744
  64.252.132.150(metric 63242) from 129.34.8.105 (129.34.8.105)
    Origin IGP, metric 0, localpref 100, valid, internal, best
    Community: no-export
```

```
mmz-b-rtr#
```

EIGRP

Border VRF

The following lines instruct the EIGRP process running inside the Border VRF to accept ("redistribute" in Cisco-speak) routes from the route table which were learned from BGP,

filtering that (enormous) list through the 'accept-gateway-of-last-resort' route-map. In the end, the only(BGP-derived) route 'redistributed' from the route table into EIGRP is 0.0.0.0.

```
! Redistribute BGP-derived routes from BGP process 14954 into EIGRP, filtering them
! through the 'accept-gateway-of-last-resort' route-map
router eigrp 12
  address-family ipv4 vrf Border
  redistribute bgp 14954 route-map accept-gateway-of-last-resort
  !
! Accept (permit) any routes matching the 'gateway-of-last-resort' ACL. Set the EIGRP metric
! for these routes: without this statement, EIGRP silent drops the routes
route-map accept-gateway-of-last-resort permit 10
  description *** Must: redistribute 0.0.0.0 from BGP into EIGRP
  match ip address gateway-of-last-resort
  set metric 1000000 1 255 1 1500
  !
! Throw away routes which match the 'all-routes' ACL
route-map accept-gateway-of-last-resort deny 20
  description *** Must: discard all other routes
  match ip address all-routes
  !
! Match everything
ip access-list standard all-routes
  remark *** Multiple Uses: Apply to all routes
  permit any
  deny any
! Match only the 0.0.0.0 route
ip access-list standard gateway-of-last-resort
  remark *** Must: accept the gateway-of-last-resort
  permit 0.0.0.0
  deny any
```

Notice how the route table tells us that it has learned this route via the "BGP 14954" process and that it is advertising ("redistributing" in Cisco-speak) this route via the "EIGRP 12" process.

```
mmz-a-rtr#sh ip ro vrf Border 0.0.0.0
Routing entry for 0.0.0.0/0, supernet
  Known via "bgp 14954", distance 200, metric 0, candidate default path
  Tag 14744, type internal
  Redistributing via eigrp 12
  Advertised by eigrp 12 route-map accept-gateway-of-last-resort
  Last update from 64.252.132.15001:10:52 ago
  Routing Descriptor Blocks:
  * 63.251.162.149, from 129.34.8.105, 01:10:52 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
    Route tag 14744
```

```
mmz-a-rtr#
```

```
mmz-b-rtr#sh ip ro vrf Border 0.0.0.0
```

```
Routing entry for 0.0.0.0/0, supernet
  Known via "bgp 14954", distance 200, metric 0, candidate default path
  Tag 14744, type internal
  Redistributing via eigrp 12
  Advertised by eigrp 12 route-map accept-gateway-of-last-resort
  Last update from 64.252.132.15000:58:00 ago
  Routing Descriptor Blocks:
  * 63.251.162.149, from 129.34.8.105, 00:58:00 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
    Route tag 14744
```

```
mmz-b-rtr#
```

If we then consult the EIGRP process running in the Border VRF, we can examine how EIGRP has received this route.

```
mmz-a-rtr#sh ip ei vrf Border top 0.0.0.0
IP-EIGRP (AS 106): Topology entry for 0.0.0.0/0
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 2816
  Routing Descriptor Blocks:
  63.251.162.149, from Redistributed, Send flag is 0x0
    Composite metric is (2816/0), Route is External
    Vector metric:
      Minimum bandwidth is 1000000 Kbit
      Total delay is 10 microseconds
      Reliability is 255/255
      Load is 1/255
      Minimum MTU is 1500
      Hop count is 0
    External data:
      Originating router is 129.34.8.101 (this system)
      AS number of route is 14954
      External protocol is BGP, external metric is 0
      Administrator tag is 14744 (0x00003998)
      Exterior flag is set
```

```
mmz-a-rtr#
```

```

mmz-b-rtr#sh ip ei vrf Border top 0.0.0.0
IP-EIGRP (AS 106): Topology entry for 0.0.0.0/0
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 2816
  Routing Descriptor Blocks:
    63.251.162.149, from Redistributed, Send flag is 0x0
      Composite metric is (2816/0), Route is External
      Vector metric:
        Minimum bandwidth is 1000000 Kbit
        Total delay is 10 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 0
      External data:
        Originating router is 129.34.8.101 (this system)
        AS number of route is 14954
        External protocol is BGP, external metric is 0
        Administrator tag is 14744 (0x00003998)
        Exterior flag is set
mmz-b-rtr#

```

Native VRF

Finally, we can examine how EIGRP in the Native VRF hears about 0.0.0.0, along with its willingness to advertise it (via the EIGRP 106 process):

```

mmz-a-rtr#sh ip ro 0.0.0.0
Routing entry for 0.0.0.0/0, supernet
  Known via "eigrp 106", distance 170, metric 3072, candidate default path
  Tag 14744, type external
  Redistributing via eigrp 106
  Last update from 129.34.8.3 on Vlan307, 01:09:47 ago
  Routing Descriptor Blocks:
  * 129.34.8.3, from 129.34.8.3, 01:09:47 ago, via Vlan307
    Route metric is 3072, traffic share count is 1
    Total delay is 20 microseconds, minimum bandwidth is 1000000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1
    Route tag 14744
mmz-a-rtr#

```

```
mmz-b-rtr#sh ip ro 0.0.0.0
Routing entry for 0.0.0.0/0, supernet
  Known via "eigrp 106", distance 170, metric 3072, candidate default path
  Tag 14744, type external
Redistributing via eigrp 106
  Last update from 129.34.9.3 on Vlan307, 01:09:51 ago
  Routing Descriptor Blocks:
  * 129.34.9.3, from 129.34.9.3, 01:09:51 ago, via Vlan307
    Route metric is 3072, traffic share count is 1
    Total delay is 20 microseconds, minimum bandwidth is 1000000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1
    Route tag 14744
```

```
mmz-b-rtr#
```

And, if we examine the Native VRF's EIGRP topology map, we can see how it will fall back to using its Native VRF EIGRP partner (*mmz-a-rtr* or *mmz-b-rtr*) via Vlan400, in the event that it loses its more direct path to the Native VRF.

```
mmz-a-rtr#sh ip ei top 0.0.0.0
IP-EIGRP (AS 106): Topology entry for 0.0.0.0/0
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 3072
  Routing Descriptor Blocks:
  129.34.8.3 (Vlan307), from 129.34.8.3, Send flag is 0x0
    Composite metric is (3072/2816), Route is External
    Vector metric:
      Minimum bandwidth is 1000000 Kbit
      Total delay is 20 microseconds
      Reliability is 255/255
      Load is 1/255
      Minimum MTU is 1500
      Hop count is 1
    External data:
      Originating router is 129.34.8.101
      AS number of route is 14954
      External protocol is BGP, external metric is 0
      Administrator tag is 14744 (0x00003998)
      Exterior flag is set
  129.34.0.41 (Vlan400), from 129.34.0.41, Send flag is 0x0
    Composite metric is (3338/3082), Route is External
    Vector metric:
      Minimum bandwidth is 1000000 Kbit
      Total delay is 30 microseconds
      Reliability is 255/255
      Load is 1/255
      Minimum MTU is 1500
      Hop count is 2
    External data:
      Originating router is 129.34.8.101
      AS number of route is 14954
      External protocol is BGP, external metric is 0
      Administrator tag is 14744 (0x00003998)
```

```

    Exterior flag is set
% IP-EIGRP (AS 12): Route not in topology table
mmz-a-rtr#

mmz-b-rtr#sh ip ei top 0.0.0.0
IP-EIGRP (AS 106): Topology entry for 0.0.0.0/0
State is Passive, Query origin flag is 1, 1 Successor(s), FD is 3072
Routing Descriptor Blocks:
 129.34.9.3 (Vlan307), from 129.34.9.3, Send flag is 0x0
   Composite metric is (3072/2816), Route is External
   Vector metric:
     Minimum bandwidth is 1000000 Kbit
     Total delay is 20 microseconds
     Reliability is 255/255
     Load is 1/255
     Minimum MTU is 1500
     Hop count is 1
   External data:
     Originating router is 129.34.8.101
     AS number of route is 14954
     External protocol is BGP, external metric is 0
     Administrator tag is 14744 (0x00003998)
     Exterior flag is set
 129.34.0.40 (Vlan400), from 129.34.0.40, Send flag is 0x0
   Composite metric is (3338/3082), Route is External
   Vector metric:
     Minimum bandwidth is 1000000 Kbit
     Total delay is 30 microseconds
     Reliability is 255/255
     Load is 1/255
     Minimum MTU is 1500
     Hop count is 2
   External data:
     Originating router is 129.34.8.101
     AS number of route is 14954
     External protocol is BGP, external metric is 0
     Administrator tag is 14744 (0x00003998)
     Exterior flag is set
% IP-EIGRP (AS 12): Route not in topology table
mmz-b-rtr#

```

High-Availability

The two halves of the MMZ, the 'a' side and the 'b' side, communicate via an 802.3ad (LACP, aka EtherChannel) pipe running between *mmz-x-rtr*. The 802.1q trunk overlaid across this channel includes:

- the Border VRF VLAN 401, carrying Border VRF EIGRP and iBGP traffic
- the Native VRF VLAN 400 carrying Native EIGRP traffic (including the reverse-route injected routes from the crypto map pushing traffic toward the IPsec SPA

modules. This VLAN also carries HSRP traffic in support of the highly-available IPsec SPA modules

- the Hutch VLAN 301 carrying VRRP traffic between the highly-available Hutch firewalls
- the GADGETS VLAN 302 carrying VRRP traffic between the highly-available GADGETS firewalls
- the 'charon' VLAN 310 carrying VRRP traffic between the highly available Hutch Remote Access VPN servers

This channel sits on top of two Gigabit Ethernet paths, each traversing the campus' backbone fiber optic ring in opposite directions.

Load Balanced vs Active/Standby

In general, we prefer simplicity over performance.

BORDER VRF

However, amongst the Edge Routers and the Border VRF, we allow both the EGP and the IGP to pick routes per their default metrics; we don't bother to influence this in an administrative way.

By default, our IGP load-balances across equal-cost routes. For example, incoming traffic crossing *internap-a-rtr* sees two paths to WIDGETS, one via *mmz-a-rtr* and another via *mmz-b-rtr*, and therefore it will load-balance (by IP address pair) across these two paths. This is more complicated than we would prefer. However, modifying this behavior requires work – more complexity in the Edge Router and Border VRF config files; therefore, by doing nothing in this regard we believe that we are sticking to our preference for simplicity.

By default, our EGP picks a single best path and uses it. For example, outbound traffic destined to *nih.gov* may see four possible routes – one across each of *internap-x-rtr* and *gigapop-x-rtr*. We let our EGP pick its favorite path and stick to it, based on the default algorithm employed by the vendor and specified in the relevant RFCs.

NATIVE VRF

Within the Native VRF, we use an Active/Standby approach; we do this partially by necessity and partially by philosophy. The IPsec modules do not support anything but an Active/Standby configuration. While the remaining gear in the Native VRF support Load Balanced modes (aka Active/Active), we choose to stick with Active/Standby, figuring that these are complex devices and that configuring them Active/Standby reduces the complexity of what we are asking them to do (their stateful exchanges don't have to be quite as close to real time). Furthermore, Active/Standby simplifies trouble-shooting in a variety of ways, from visualizing traffic flow to capturing packets to managing upgrades.⁷

⁷ Upgrade the Standby node, fail over to it, test, if testing uncovers problems, fail back to the previously (and non-upgraded) Active node.

Thus, the IPsec module in *mmz-a-rtr* is normally Active, whereas its partner in *mmz-b-rtr* is normally Standby. Ditto for the HSRP sides of *mmz-x-rtr*, *charon-x-vpn*, *ice-x-fw*, and *ga-x-fw*.

VALIDATION

Currently, we verify the highly-available functionality in the MMZ by rebooting each device in turn, during a scheduled outage window, and recording the results of pings traversing the MMZ. After each test, the gear is configured to automatically fail-back to its usual Active/Standby arrangement, where the 'a' side is Active.

UNDERSTANDING THE IPSEC SPA MODULES

Consult the following resources to understand how these modules function.

See `\\indigo\admshred\it\vdops\tech-dox\miscellaneous\kkawakub\site-to-site-vpn` for Ken's documentation on the original IPsec implementation; this contains a concise description of how the IPsec modules work.

The SPA module is a 'carrier' card -- it slides into a C6K and sucks power. Typically, one inserts one or more sub-modules into the carrier card; in our case, we install an IPsec encryption off-load engine, capable of 2Gb/s of throughput. It functions as a 'bump-in-the-wire', sitting somewhere between a repeater (Layer 1) and a bridge (Layer 2) in the OSI model.

Cisco aimed this IPsec module at the carrier market -- a C6K can host 9? 10? of these cards. The customer dedicates one or more C6K to these cards and inserts them in-line with the routed traffic to be encrypted/decrypted. Notice the narrowness of the market sector and the narrowness of the deployment scenario -- this is not a general purpose product trying to meet the needs of a range of customers. Rather, it is a focussed product, aimed at a niche in one market which prizes performance over flexibility and ease-of-use.

We haven't been willing to dedicate a C6K to these modules. Instead, we have integrated them into our MMZ's Layer 3 core. We use VRF-Lite as a "poor person's" way to 'dedicate' a C6K to each module.

Configuring VPNs in Crypto Connect Mode

http://www.cisco.com/en/US/docs/interfaces_modules/shared_port_adapters/configuration/6500series/76cfvpn1.html#wp2024421

Switch Outside Ports and Inside Ports

The Fast Ethernet or Gigabit Ethernet ports on the Catalyst 6500 series switch that connect to the WAN routers are referred to as switch outside ports. These ports connect

the LAN to the Internet or to remote sites. Cryptographic policies are applied to the switch outside ports.

The Fast Ethernet or Gigabit Ethernet ports on the Catalyst 6500 series switch that connect to the LAN are referred to as switch inside ports.

The IPsec VPN SPA sends encrypted packets to the switch outside ports and decrypted packets to the Policy Feature Card (PFC) for Layer 3 forwarding to the switch inside ports.

IPsec VPN SPA Outside Port and Inside Port

The IPsec VPN SPA appears to the CLI as a SPA with two Gigabit Ethernet ports. The IPsec VPN SPA has no external connectors; the Gigabit Ethernet ports connect the IPsec VPN SPA to the switch backplane and Switch Fabric Module (SFM) (if installed).

Port VLAN and Interface VLAN

Your VPN configuration can have one or more switch outside ports. To handle the packets from multiple switch outside ports, you must direct the packets from multiple switch outside ports to the IPsec VPN SPA outside port by placing the switch outside ports in a VLAN with the outside port of the IPsec VPN SPA. This VLAN is referred to as the port VLAN. The port VLAN is a Layer 2-only VLAN. You do not configure Layer 3 addresses or features on this VLAN; the packets within the port VLAN are bridged by the PFC.

Before the switch can forward the packets using the correct routing table entries, the switch needs to know which interface a packet was received on. For each port VLAN, you must create another VLAN so that the packets from every switch outside port are presented to the switch with the corresponding VLAN ID. This VLAN contains only the IPsec VPN SPA inside port and is referred to as the interface VLAN. The interface VLAN is a Layer 3-only VLAN. You configure the Layer 3 address and Layer 3 features, such as ACLs and the crypto map, to the interface VLAN.

You tie the port VLAN and the interface VLAN together using the crypto engine slot command on the interface VLAN followed by the crypto connect vlan command on the port VLAN. Figure 26-1 shows an example of the port VLAN and interface VLAN configurations.

One Gigabit Ethernet port handles all the traffic going to and coming from the switch outside ports. This port is referred to as the IPsec VPN SPA outside port. The other Gigabit Ethernet port handles all traffic going to and coming from the LAN or switch inside ports. This port is referred to as the IPsec VPN SPA inside port.

IPsec LAN-to-LAN Tunnel Between a Catalyst 6500 with the VPN Service Module and a Cisco IOS Router Configuration Example

http://www.cisco.com/en/US/tech/tk583/tk372/technologies_configuration_example09186a00800f6d82.shtml

The Catalyst 6500 VPN service module has two Gigabit Ethernet (GE) ports with no externally visible connectors. These ports are addressable for configuration purposes only. Port 1 is always the inside port. This port handles all traffic from and to the inside network. The second port (port 2) handles all traffic from and to the WAN or outside networks. These two ports are always configured in 802.1Q trunking mode. The VPN service module uses a technique called Bump In The Wire (BITW) for packet flow.

*Packets are processed by a pair of VLANs, one Layer 3 inside VLAN and one Layer 2 outside VLAN. The packets, from the inside to the outside, are routed through a method called Encoded Address Recognition Logic (EARL) to the inside VLAN. After it encrypts the packets, the VPN service module uses the corresponding outside VLAN. In the decryption process, the packets from the outside to the inside are bridged to the VPN service module using the outside VLAN. After the VPN service module decrypts the packet and maps the VLAN to the corresponding inside VLAN, EARL routes the packet to the appropriate LAN port. The Layer 3 inside VLAN and the Layer 2 outside VLANs are joined together by issuing the **crypto connect vlan** command.*

Deploying the IPsec SPA module requires pushing traffic which you want encrypted from the 'inside' VLAN to the 'outside' VLAN. And pushing traffic which you want decrypted from the 'outside' VLAN to the 'inside' VLAN. The IPsec SPA module sits in between the 'inside' and 'outside' VLANs, decrypting/encrypting whatever passes through it.

VLAN 303 is the 'inside' VLAN (aka the 'interface VLAN') and VLAN 305 is the 'outside' VLAN (aka the 'port VLAN').

IPsec Stateful Failover (VPN High Availability) Feature Module

http://www.cisco.com/en/US/customer/docs/ios/12_2/12_2y/12_2yx11/feature/guide/ft_vpnha.html

Feature Overview

IPSec Stateful Failover (VPN High Availability) is a feature that enables a router to continue processing and forwarding packets after a planned or unplanned outage. You can employ a backup (standby) router that automatically takes over the primary (active) router's tasks in the event of an active router failure. The process is transparent to users and to remote IPsec peers. The time that it takes for the standby router to take over depends on HSRP timers.

IPSec Stateful Failover (VPN High Availability) is designed to work in conjunction with Reverse Route Injection (RRI) and Hot Standby Router Protocol (HSRP) with IPSec. When used together, RRI and HSRP provide a more reliable network design for VPNs and reduce configuration complexity on remote peers.

RRI and HSRP are supported together with the restriction that the HSRP configuration on the outside interface uses equal priorities on both routers. As an option, when not using RRI, you can use an HSRP configuration on the LAN side of the network (equal HSRP priority restriction still applies).

Configuring Duplicate Hardware and IPsec Failover Using the IPsec VPN SPA

http://www.cisco.com/en/US/docs/interfaces_modules/shared_port_adapters/configuration/6500series/76cfvpn6.html

Understanding Stateful Failover Using HSRP and SSP

Note: Support for IPsec stateful failover using HSRP and SSP is removed in Cisco IOS Release 12.2(33)SXH and later releases. The feature is supported in Release 12.2SXF.

IPsec stateful failover enables a switch to continue processing and forwarding IPsec packets after a planned or unplanned outage. The failover process is transparent to users and to remote IPsec peers.

As with IPsec stateless failover, IPsec stateful failover is designed to work with HSRP and RRI, but IPsec stateful failover also uses the State Synchronization Protocol (SSP). During an HSRP and IPsec failover, SSP transfers IPsec and ISAKMP SA state information between the active and standby switches, allowing existing VPN connections to be maintained after a switch failover.

Troubleshooting the IPsec VPN SPA

http://www.cisco.com/en/US/customer/docs/interfaces_modules/shared_port_adapters/configuration/6500series/76tblvpn.html#wp1076127

EXPLORE TUNNELS

Examine Hardware

With these commands, we look at the off-load hardware.

show crypto engine brief

Verifies that the chassis sees the IPsec off-load engine; use this to look for hardware problems or OS incompatibilities.

```
mmz-a-rtr#sh crypto engine br
```

```
crypto engine name: Virtual Private Network (VPN) Module
crypto engine type: hardware
  Compression: No
    DES: Yes
    3 DES: Yes
    AES CBC: Yes (128,192,256)
    AES CNTR: No
Maximum buffer length: 1492
  Maximum DH index: 9999
  Maximum SA index: 10921
  Maximum Flow index: 21842
  Maximum RSA key size: 0000
crypto engine in slot: 5
```

```
crypto engine name: unknown
crypto engine type: software
  serial number: 00000000
crypto engine state: installed
crypto engine in slot: N/A
```

```
mmz-a-rtr#
```

```
mmz-b-rtr#sh crypto engine brief
```

```
crypto engine name: Virtual Private Network (VPN) Module
crypto engine type: hardware
  Compression: No
    DES: Yes
    3 DES: Yes
    AES CBC: Yes (128,192,256)
    AES CNTR: No
Maximum buffer length: 1492
  Maximum DH index: 9999
  Maximum SA index: 10921
  Maximum Flow index: 21842
  Maximum RSA key size: 0000
crypto engine in slot: 5
```

```
crypto engine name: unknown
crypto engine type: software
  serial number: 00000000
crypto engine state: installed
crypto engine in slot: N/A
```

```
mmz-b-rtr#
```

show crypto eli

Displays summary statistics about how much work the IPsec off-load engine is performing. Use this to verify that the IOS is using the IPsec module to off-load crypto work (rather than performing it using the Sup card's CPU).

```
mmz-a-rtr#sh crypto eli
Hardware Encryption Layer :    ACTIVE
Number of crypto engines = 1 .

CryptoEngine-SPA-IPSEC-2G[5/0] (slot-5/0) details.
Capability-IPSec : No-IPPCP, 3DES, AES, RSA

IKE-Session   :      9 active, 10921 max, 0 failed
DH-Key        :      0 active,  9999 max, 0 failed
IPSec-Session :     42 active, 21842 max, 0 failed
```

mmz-a-rtr#

```
mmz-b-rtr#sh crypto eli
Hardware Encryption Layer :    ACTIVE
Number of crypto engines = 1 .

CryptoEngine-SPA-IPSEC-2G[5/0] (slot-5/0) details.
Capability-IPSec : No-IPPCP, 3DES, AES, RSA

IKE-Session   :      0 active, 10921 max, 0 failed
DH-Key        :      0 active,  9999 max, 0 failed
IPSec-Session :      0 active, 21842 max, 0 failed
```

mmz-b-rtr#

Basic Health

show crypto vlan

Verifies that the 'inside' and 'outside' interfaces of the IPsec off-load engine is correctly assigned to VLANs.

```
mmz-a-rtr#sh crypto vlan
Interface VLAN 303 on IPsec Service Module port Gi5/0/1 connected to VLAN 305
with crypto map set gadgetsvpn
```

mmz-a-rtr#

```
mmz-b-rtr#sh crypto vlan
Interface VLAN 303 on IPsec Service Module port Gi5/0/1 connected to VLAN 305
with crypto map set gadgetsvpn
```

mmz-b-rtr#

show-tunnels

This script displays the status of the tunnels. Use this when you want to see rapidly which routers are hosting which tunnels.⁸

```
jacobsite> show-tunnels
Beginning /home/netops/bin/show-tunnels
Gathering status...
```

```
Pinging host list...
!!
```

```
Characterizing host list...
!!
!!
!!
```

```
Acquiring tunnel information...
!
```

```
# Monday December 01, 2008 at 06:06:08
# target      peer                active time                traffic
# -----
-
  mmz-a-rtr   icad-vpn            1 minute and 59 seconds    4 KB
              mcis-prod           2 minutes and 52 seconds    6 KB
              impac-vpn           2 minutes and 32 seconds    2 KB
              mcis-prod-mimi3     2 minutes and 54 seconds    2 KB
              mcis-prod-mimi3     2 minutes and 52 seconds    3 KB
              sod             2 minutes and 45 seconds    2 KB
              gems            2 minutes and 45 seconds    2 KB
              chrmc          2 minutes and 45 seconds    4 KB
              pms-vpn         1 minute and 59 seconds    5 KB

  mmz-b-rtr
```

```
Ending /home/netops/bin/show-tunnels
jacobsite>
```

Crypto Status in Detail

MMZ-A-RTR

show crypto isakmp sa

List the currently active tunnels.

```
mmz-a-rtr#sh crypto isakmp sa
```

⁸ This script emulates the output of `show crypto isakmp sa`.

dst	src	state	conn-id	slot
64.240.36.31	129.34.8.201	QM_IDLE	10	0
144.122.68.117	129.34.8.201	QM_IDLE	5	0
108.52.41.240	129.34.8.201	QM_IDLE	8	0
128.65.121.225	129.34.8.201	QM_IDLE	9	0
206.160.213.80	129.34.8.201	QM_IDLE	11	0
129.34.8.201	219.125.180.60	QM_IDLE	7	0
129.34.8.201	126.217.166.6	QM_IDLE	3	0
129.34.8.201	126.217.166.6	QM_IDLE	2	0
129.34.8.201	65.94.121.76	QM_IDLE	4	0

mmz-a-rtr#

show crypto isakmp peers

Display detailed information about each tunnel peer.

mmz-a-rtr#sh crypto isakmp peers

Peer: 64.240.36.31 Port: 500

Phase1 id: 64.240.36.31

Peer Index: 0

flags: DPD information, struct 0x4AF3BB58:

Last_received: 7, dpd threshold (elapsed) 0

my_last_seq_num: 0x0, peers_last_seq_num: 0x13B62FCE

sent_and_waiting: FALSE

Peer: 65.94.121.76 Port: 500

Phase1 id: 65.94.121.76

Peer Index: 0

flags:

Peer: 128.65.121.225 Port: 500

Phase1 id: 128.65.121.225

Peer Index: 0

flags: DPD information, struct 0x5C319918:

Last_received: 310, dpd threshold (elapsed) 0

my_last_seq_num: 0x0, peers_last_seq_num: 0x0

sent_and_waiting: FALSE

Peer: 126.217.166.6 Port: 500

Phase1 id: 126.217.166.6

Peer Index: 0

flags:

Peer: 144.122.68.117 Port: 500

Phase1 id: 144.122.68.117

Peer Index: 0

flags: DPD information, struct 0x526D2788:

Last_received: 325, dpd threshold (elapsed) 0

my_last_seq_num: 0x0, peers_last_seq_num: 0x0

sent_and_waiting: FALSE

Peer: 108.52.41.240 Port: 500

Phase1 id: 108.52.41.240

Peer Index: 0

flags: DPD information, struct 0x7AE14510:

InfoTech/Admin/FHCRC

27

Stuart Kendrick

2009-01-18
MMZ Design

```
Last_received: 324, dpd threshold (elapsed) 0
my_last_seq_num: 0x0, peers_last_seq_num: 0x0
sent_and_waiting: FALSE
```

```
Peer: 219.125.180.60 Port: 500
Phase1 id: 219.125.180.60
Peer Index: 0
flags: DPD information, struct 0x56956150:
Last_received: 325, dpd threshold (elapsed) 0
my_last_seq_num: 0x0, peers_last_seq_num: 0x0
sent_and_waiting: FALSE
```

```
Peer: 206.160.213.80 Port: 500
Phase1 id: 206.160.213.80
Peer Index: 0
flags: DPD information, struct 0x4AF3BCB8:
Last_received: 1, dpd threshold (elapsed) 0
my_last_seq_num: 0x0, peers_last_seq_num: 0x45397FF2
sent_and_waiting: FALSE
```

```
mmz-a-rtr#
```

MMZ-B-RTR

```
show crypto isakmp sa
```

```
mmz-b-rtr#
```

```
mmz-b-rtr#sh crypto isakmp sa
dst          src          state          conn-id slot
```

```
mmz-b-rtr#
```

```
show crypto isakmp peers
```

```
mmz-b-rtr#sh crypto isakmp peers
```

```
mmz-b-rtr#
```

Stateful Synchronization Protocol

We employ SSP to push state (as far as I can tell, this consists of ISAKMP keys) from the Active module to the Standby module, to reduce failover time.

We enable SSP using the following commands. '25' is the SSP group number and is a number I pulled from thin air.

```
mmz-a-rtr
ssp group 25
remote 129.34.8.204
redundancy spa-hsrp
!
```

```
crypto isakmp ssp 25
!
interface Vlan303
 standby 6 name spa-hsrp
 crypto map gadgetsvpn ssp 25
```

```
mmz-b-rtr
ssp group 25
 remote 129.34.8.203
 redundancy spa-hsrp
!
crypto isakmp ssp 25
!
interface Vlan303
 standby 6 name spa-hsrp
 crypto map gadgetsvpn ssp 25
```

show ssp

This command displays the status of this node (ACTIVE or STANDBY), the HSRP groups identifying the interfaces carrying traffic to be encrypted, and the status of this node's peers. Use this to check that the two SSP partners see each other and are communicating. Use the show ssp client, show ssp peer, show ssp redundancy, and show ssp packet commands to view this output in chunks. Generally, I use the sh ssp peer command; I haven't found a use for the rest of the output.

```
mmz-a-rtr#sh ssp
SSP Client Information
  DOI   Client Name           Version   Running Ver
   2    IKE HA Manager         1.0      1.0
   1    IPsec HA Manager       1.0      1.0
SSP Redundancy Information
Device has been ACTIVE for 1d06h
Virtual IP      Redundancy Name      Interface
129.34.0.1     Widgets-hsrp         Vlan301
129.34.0.17    gadgets-hsrp         Vlan302
129.34.8.201   spa-hsrp             Vlan303
SSP Peer Information
IP Address      Connection State     Local Interface
129.34.8.204   Connected            Vlan303
SSP packet Information
Socket creation time: 1d06h
Local port: 3249      Server port: 3249
Packets Sent = 10636, Bytes Sent = 1547866
Packets Received = 20, Bytes Received = 680
mmz-a-rtr#
```

```
mmz-b-rtr#sh ssp
SSP Client Information
  DOI   Client Name           Version   Running Ver
   2    IKE HA Manager         1.0      1.0
   1    IPsec HA Manager       1.0      1.0
```

SSP Redundancy Information

Device has been STANDBY for 1d06h

Virtual IP	Redundancy Name	Interface
129.34.0.1	Widgets-hsrp	Vlan301
129.34.0.17	gadgets-hsrp	Vlan302
129.34.8.201	spa-hsrp	Vlan303

SSP Peer Information

IP Address	Connection State	Local Interface
129.34.8.203	Connected	Vlan303

SSP packet Information

Socket creation time: 1d06h
Local port: 28092 Server port: 3249
Packets Sent = 20, Bytes Sent = 680
Packets Received = 10636, Bytes Received = 1547866

mmz-b-rtr#

show crypto isakmp ha

This command lists the HSRP groups for which the SSP protocol is propagating state. It also shows whether this node is Active or Standby for that particular HSRP group.

```
mmz-a-rtr#sh crypto isakmp ha
VIP          SAs      Stamp      HA State
129.34.8.201  8        AD3F418    Active since 07:39:41 pst Sat Jan 3 2009
```

mmz-a-rtr#

```
mmz-b-rtr#sh crypto isakmp ha
VIP          SAs      Stamp      HA State
129.34.8.201  8        AD3F418    Standby since 07:52:38 pst Sat Jan 3 2009
```

mmz-b-rtr#

show crypto isakmp ha active

This command lists the ISAKMP key pairs for which this node is using SSP to propagate state. Use it to verify that your favorite tunnel is covered by SSP.

```
mmz-a-rtr#sh crypto isakmp ha active
dst          src          state        I-Cookie      R-Cookie
129.34.8.201 126.217.166.6 QM_IDLE      86D25946 A4FBF453 E20989CC 8900D9C8
129.34.8.201 126.217.166.6 QM_IDLE      0C4D5C10 21C4DBB2 E20989CC 52ADC2CB
206.160.213.80 129.34.8.201 QM_IDLE      DB4A53D4 E0B51F45 E8D3327F 3BBBC97B
108.52.41.240 129.34.8.201 QM_IDLE      D7AA0ECF FAB4FF6D A2D4AEA0 7A1F4DD2
128.65.121.225 129.34.8.201 QM_IDLE      D573CB38 EBB4240 4D2B4FC0 0CF9E9F2
129.34.8.201 219.125.180.60 QM_IDLE      258C9C2F 482DEF4B 6AF900E1 9AF7EFED
129.34.8.201 64.240.36.31 QM_IDLE      B4D9A6C7 9312F3CF 429015DB 639236C8
144.122.68.117 129.34.8.201 QM_IDLE      34C107E7 DAD04451 1679F252 FF40C19A
129.34.8.201 65.94.121.76 QM_IDLE      A0B50EF7 8F1B4508 0BE91536 5F3D38C0
```

mmz-a-rtr#

```
mmz-a-rtr#sh crypto isakmp ha stand
dst          src          state        I-Cookie      R-Cookie
```

mmz-a-rtr#

show crypto isakmp ha standby

This command lists the ISAKMP key pairs for which this node is receiving state via SSP. Use it to verify that your favorite tunnel is covered by SSP.

```
mmz-b-rtr#sh crypto isakmp ha active
dst          src          state      I-Cookie      R-Cookie

mmz-b-rtr#

mmz-b-rtr#sh crypto isakmp ha standby
dst          src          state      I-Cookie      R-Cookie
129.34.8.201 126.217.166.6 QM_IDLE   86D25946 A4FBF453 E20989CC 8900D9C8
129.34.8.201 126.217.166.6 QM_IDLE   0C4D5C10 21C4DBB2 E20989CC 52ADC2CB
206.160.213.80 129.34.8.201 QM_IDLE   DB4A53D4 E0B51F45 E8D3327F 3BBBC97B
108.52.41.240 129.34.8.201 QM_IDLE   D7AA0ECF FAB4FF6D A2D4AEA0 7A1F4DD2
128.65.121.225 129.34.8.201 QM_IDLE   D573CB38 EBBC4240 4D2B4FC0 0CF9E9F2
129.34.8.201 219.125.180.60 QM_IDLE   258C9C2F 482DEF4B 6AF900E1 9AF7EFED
129.34.8.201 64.240.36.31 QM_IDLE   B4D9A6C7 9312F3CF 429015DB 639236C8
144.122.68.117 129.34.8.201 QM_IDLE   34C107E7 DAD04451 1679F252 FF40C19A
129.34.8.201 65.94.121.76 QM_IDLE   A0B50EF7 8F1B4508 0BE91536 5F3D38C0

mmz-b-rtr#
```

show crypto isakmp ha counters

I haven't found a use for this, but the TAC asked for it at one point, so I document it here.

```
mmz-a-rtr#sh crypto isakmp ha counters
IKE HA internal counters
~~~~~
SA HA Create.....14
SA HA Create RETRY.....0
SA HA Create SUCCESS.....0
SA HA Create QUEUE FULL..0
SA HA Create FAIL.....0
SA HA Query.....62
SA HA Query RETRY.....0
SA HA Query SUCCESS.....62
SA HA Query QUEUE FULL...0
SA HA Query SA DELETED...0
SA HA Query FAIL.....0
mmz-a-rtr#
```

```
mmz-b-rtr#sh crypto isakmp ha counters
IKE HA internal counters
~~~~~
SA HA Create.....49
SA HA Create RETRY.....0
SA HA Create SUCCESS.....0
SA HA Create QUEUE FULL..0
SA HA Create FAIL.....0
SA HA Query.....0
SA HA Query RETRY.....0
SA HA Query SUCCESS.....0
SA HA Query QUEUE FULL...0
```

```
SA HA Query SA DELETED...0
SA HA Query FAIL.....0
mmz-b-rtr#
```

Crypto and SSP Syslog Messages

NORMAL

During normal operation, the Active SSP node intermittently sends the following to syslog. This emanates from a debug routine accidentally left enabled and should go away in the next code release.

```
Jan 18 07:26:27 mmz-a-rtr-native 603: 000518: Jan 18 07:26:30 pst:
IPSEC_FLOW_QUERY KEYS.
```

FAILOVER

Here is what you should see in syslog during failover (in this case, *mmz-a-rtr* is rebooting). I don't know why some messages are stuttered.

```
Jan 18 07:26:37 mmz-a-rtr-native 611: 000526: Jan 18 07:26:41 pst: %SSP-5-
DISABLED: SSP entering disabled state.
```

```
Jan 18 07:26:37 mmz-b-rtr-native 353: 000278: Jan 18 07:26:41 pst: %SSP-6-
ACTIVE: SSP entering active state.
```

```
Jan 18 07:26:37 mmz-b-rtr-native 354: 000279: Jan 18 07:26:41 pst:
%CRYPTO_HA-6-IKEFAILOVER: (VIP=129.34.8.201)Taking over as the new Active
device for the ISAKMP failover group.
```

```
Jan 18 07:26:37 mmz-b-rtr-native 355: 000280: Jan 18 07:26:41 pst:
%CRYPTO_HA-6-IPSECFILOVER: (VIP=129.34.8.201)Taking over as the new Active
device for the IPSEC failover group.
```

```
Jan 18 07:26:37 mmz-a-rtr-native 612: 000527: Jan 18 07:26:41 pst:
%CRYPTO_HA-6-IKEDOWN: (VIP=129.34.8.201)Disabling High Availability
functionality for the ISAKMP failover group.
```

```
Jan 18 07:26:37 mmz-a-rtr-native 613: 000528: Jan 18 07:26:41 pst:
%CRYPTO_HA-6-IPSECDOWN: (VIP=129.34.8.201)Disabling High Availability
functionality for the IPSEC failover group.
```

Now that *mmz-b-rtr* is the Active SSP node, it starts emitting these 'normal' messages.

```
Jan 18 07:27:46 mmz-b-rtr-native 417: 000338: Jan 18 07:27:50 pst:
IPSEC_FLOW_QUERY KEYS.
```

FAILBACK

As *mmz-a-rtr* returns to life, it logs the following POST messages related to the IPsec module.

```
Jan 18 07:31:23 mmz-a-rtr-native 331: .Jan 18 07:31:27 pst%SPA-IPSEC-2G-6-
FIPS582XNOTIFY: slot 5/0/1 FipsRNG POST: Successful (0)
Jan 18 07:31:23 mmz-a-rtr-native 332: .Jan 18 07:31:27 pst%SPA-IPSEC-2G-6-
FIPS584XNOTIFY: slot 5/0/1 Fips584x POST: Successful (0)
Jan 18 07:31:23 mmz-a-rtr-native 333: .Jan 18 07:31:27 pst%SPA-IPSEC-2G-6-
FIPS582XNOTIFY: slot 5/0/1 Fips582x POST: Successful (0)
Jan 18 07:31:23 mmz-a-rtr-native 334: 000257: .Jan 18 07:31:27 pst: %LINK-3-
UPDOWN: Interface GigabitEthernet5/0/2, changed state to up
Jan 18 07:31:25 mmz-a-rtr-native 335: 000258: .Jan 18 07:31:29 pst: %ACE-6-
INFO: SPA-IPSEC-2G[5/0]: Recognised crypto engine (4)
Jan 18 07:31:25 mmz-a-rtr-native 336: 000259: .Jan 18 07:31:29 pst: %CRYPTO-
6-ISAKMP_ON_OFF: ISAKMP is OFF
Jan 18 07:31:25 mmz-a-rtr-native 337: 000260: .Jan 18 07:31:29 pst: %CRYPTO-
6-ISAKMP_ON_OFF: ISAKMP is ON
```

With the IPsec module functional, *mmz-a-rtr* brings Vlan303 on-line.

```
Jan 18 07:31:25 mmz-a-rtr-native 338: 000261: .Jan 18 07:31:29 pst: %LINK-3-
UPDOWN: Interface Vlan303, changed state to up
```

I believe this is another normal (debug) message.

```
Jan 18 07:31:32 mmz-b-rtr-native 502: 000419: Jan 18 07:31:36 pst:
IKEA_SA_QUERY_KEYS ..
```

At this point, I believe that *mmz-a-rtr* has acquired a copy of the ISAKMP keys which *mmz-b-rtr* has been maintaining, via SSP, and is now functioning as the SSP Standby node.

```
Jan 18 07:31:33 mmz-a-rtr-native 342: 000265: .Jan 18 07:31:36 pst: %SSP-6-
STANDBY: SSP entering standby state.
Jan 18 07:31:33 mmz-a-rtr-native 343: 000266: .Jan 18 07:31:36 pst:
%CRYPTO_HA-6-IKESTANDBY: (VIP=129.34.8.201)Setting up as a Standby device for
the ISAKMP failover group.
Jan 18 07:31:33 mmz-a-rtr-native 344: 000267: .Jan 18 07:31:36 pst:
%CRYPTO_HA-6-IPSECSTANDBY: (VIP=129.34.8.201)Setting up as a Standby device
for the IPSEC failover group.
Jan 18 07:31:33 mmz-a-rtr-native 345: 000268: .Jan 18 07:31:36 pst:
%CRYPTO_HA-6-IKEHASYNCCOMPLETE: (VIP=129.34.8.201)IKE HA state
synchronization with Active device complete.
Jan 18 07:31:33 mmz-a-rtr-native 346: 000269: .Jan 18 07:31:36 pst:
%CRYPTO_HA-6-IPSECHASYNCCOMPLETE: (VIP=129.34.8.201)IPSEC HA state
synchronization with Active device complete.
```

Several minutes later, *mmz-a-rtr* resumes its HSRP Active role.

```
Jan 18 07:32:49 mmz-a-rtr-native 350: 000273: Jan 18 07:32:52 pst: %STANDBY-
6-STATECHANGE: Vlan310 Group 7 state Standby -> Active
Jan 18 07:32:49 mmz-a-rtr-native 351: 000274: Jan 18 07:32:53 pst: %STANDBY-
6-STATECHANGE: Vlan301 Group 4 state Standby -> Active
```

```
Jan 18 07:32:49 mmz-a-rtr-native 352: 000275: Jan 18 07:32:53 pst: %STANDBY-6-STATECHANGE: Vlan302 Group 5 state Standby -> Active
Jan 18 07:33:32 mmz-a-rtr-native 356: 000279: Jan 18 07:33:35 pst: %STANDBY-6-STATECHANGE: Vlan303 Group 6 state Standby -> Active
```

SSP pays attention to HSRP status on Vlan303 and follows it.⁹

```
Jan 18 07:33:32 mmz-a-rtr-native 357: 000280: Jan 18 07:33:35 pst: %SSP-6-ACTIVE: SSP entering active state.
Jan 18 07:33:32 mmz-a-rtr-native 358: 000281: Jan 18 07:33:35 pst: %CRYPTO_HA-6-IKEFAILOVER: (VIP=129.34.8.201)Taking over as the new Active device for the ISAKMP failover group.
Jan 18 07:33:32 mmz-a-rtr-native 359: 000282: Jan 18 07:33:35 pst: %CRYPTO_HA-6-IPSECFAILOVER: (VIP=129.34.8.201)Taking over as the new Active device for the IPSEC failover group.
Jan 18 07:33:34 mmz-b-rtr-native 587: 000504: Jan 18 07:33:38 pst: %SSP-5-DISABLED: SSP entering disabled state.
```

And now *mmz-a-rtr* starts emitting the debug lines while *mmz-b-rtr* goes silent.

```
Jan 18 07:33:35 mmz-a-rtr-native 360: 000283: Jan 18 07:33:39 pst:
IKEA_SA_QUERY_KEYS ..
```

mmz-a-rtr is now Active, both in the HSRP and in the SSP sense, while *mmz-b-rtr* is Standby.

EXPLORE NEIGHBORS

Native VRF

show cdp neighbors

Notice that the Edge Routers appear once each – fairly predictable. The partner, *mmz-b-rtr*, appears twice: once for each of the GigE ports employed in the EtherChannel linking the two. And notice how this box, *mmz-a-rtr*, appears four (4) times: once for each of the two Native VRF ports leading to the Border VRF ... and then again once for each of the Border VRF ports.

```
mmz-a-rtr#sh cdp nei
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone

Device ID          Local Intrfce    Holdtme    Capability  Platform  Port ID
internap-a-rtr.widgets.com
                  Gig 3/6          158        R           7206VXR   Gig 0/1
internap-b-rtr.widgets.com
                  Gig 3/8          165        R           7206VXR   Gig 0/1
mmz-a-rtr.widgets.com
                  Gig 4/15         178        R S I       WS-C6506  Gig 4/16
mmz-a-rtr.widgets.com
                  Gig 4/16         178        R S I       WS-C6506  Gig 4/15
```

⁹ In other words, when *mmz-a-rtr* goes becomes HSRP Active for Vlan303, it also becomes the SSP Active node. This is part of what the 'redundancy spa-hsrp' statement under the 'ssp group 25' stanza delivers.

```

mmz-a-rtr.widgets.com
    Gig 4/1          178      R S I    WS-C6506 Gig 4/2
mmz-a-rtr.widgets.com
    Gig 4/2          178      R S I    WS-C6506 Gig 4/1
mmz-b-rtr.widgets.com
    Gig 3/7          143      R S I    WS-C6506 Gig 3/7
mmz-b-rtr.widgets.com
    Gig 3/3          143      R S I    WS-C6506 Gig 3/3
manwe               Gig 6/1          149      R S I    3845     Gig 0/0
gigapop-a-rtr.widgets.com
    Gig 3/1          143      R S I    WS-C6504-EGig 1/2
gigapop-b-rtr.widgets.com
    Gig 3/2          144      R S I    WS-C6504-EGig 1/2
mmz-a-rtr#

```

mmz-b-rtr displays the same neighbors.

```

mmz-b-rtr#sh cdp nei
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone

Device ID          Local Intrfce    Holdtme    Capability  Platform  Port ID
internap-a-rtr.widgets.com
    Gig 3/6          170         R           7206VXR    Gig 0/2
internap-b-rtr.widgets.com
    Gig 3/8          150         R           7206VXR    Gig 0/2
mmz-a-rtr.widgets.com
    Gig 3/7          167         R S I       WS-C6506   Gig 3/7
mmz-a-rtr.widgets.com
    Gig 3/3          167         R S I       WS-C6506   Gig 3/3
mmz-b-rtr.widgets.com
    Gig 4/15         163         R S I       WS-C6506   Gig 4/16
mmz-b-rtr.widgets.com
    Gig 4/16         163         R S I       WS-C6506   Gig 4/15
mmz-b-rtr.widgets.com
    Gig 4/1          163         R S I       WS-C6506   Gig 4/2
mmz-b-rtr.widgets.com
    Gig 4/2          163         R S I       WS-C6506   Gig 4/1
manwe               Gig 6/1          175         R S I       3845       Gig 0/1
gigapop-a-rtr.widgets.com
    Gig 3/1          123         R S I       WS-C6504-EGig 1/3
gigapop-b-rtr.widgets.com
    Gig 3/2          141         R S I       WS-C6504-EGig 1/3
mmz-b-rtr#

```

show ip eigrp neighbors

The Native VRF on *mmz-a-rtr* has two EIGRP neighbors:

- *mmz-b-rtr*, at 129.34.0.41, reachable via the Vlan400 interface (this VLAN traverses the interconnecting EtherChannel)
- the Border VRF of *mmz-a-rtr*, hosted across Vlan307, at 129.34.8.3

```

mmz-a-rtr#sh ip ei nei
IP-EIGRP neighbors for process 106
H   Address                Interface          Hold Uptime      SRTT   RTO  Q  Seq
                               (sec)             (ms)              Cnt  Num
1   129.34.0.41              V1400              2 03:37:24      1    200  0  15
0   129.34.8.3                V1307              2 04:05:01      6    200  0 2080
IP-EIGRP neighbors for process 12
mmz-a-rtr#

```

And *mmz-b-rtr*'s Native VRF EIGRP neighbors appear similarly.

```
mmz-b-rtr#sh ip ei nei
IP-EIGRP neighbors for process 106
H   Address                Interface      Hold Uptime    SRTT   RT0   Q   Seq
                               (sec)          (ms)          Cnt Num
1   129.34.9.3              V1307         2 05:43:53    1     200  0   87
0   129.34.0.40             V1400         2 05:43:53    82    492  0  196
IP-EIGRP neighbors for process 12
mmz-b-rtr#
```

Border VRF

show ip eigrp vrf Border neighbors

The Border VRF sees the Edge Routers as EIGRP partners.

```
mmz-a-rtr#sh ip ei vrf Border nei
IP-EIGRP neighbors for process 106
H   Address                Interface      Hold Uptime    SRTT   RT0   Q   Seq
                               (sec)          (ms)          Cnt Num
1   129.34.8.41             V1401         2 5d01h       232   1392  0   89
6   129.34.8.13             V1412         2 5d01h         1     200  0  283
5   129.34.8.19             V1418         2 5d01h         36    216  0  255
4   129.34.8.17             V1416         2 5d01h         12    200  0  242
3   129.34.8.15             V1414         2 5d01h          8     200  0  271
2   129.34.8.11             V1410         2 5d01h       316   2844  0  439
0   129.34.8.2              V1308         2 5d01h         38    228  0   34
mmz-a-rtr#
```

```
mmz-b-rtr#sh ip ei vrf Border ne
IP-EIGRP neighbors for process 106
H   Address                Interface      Hold Uptime    SRTT   RT0   Q   Seq
                               (sec)          (ms)          Cnt Num
6   129.34.9.17             V1516         2 5d01h         90    540  0  243
5   129.34.9.15             V1514         2 5d01h         46    276  0  272
4   129.34.9.19             V1518         2 5d01h         57    342  0  256
3   129.34.9.13             V1512         2 5d01h         48    288  0  284
2   129.34.8.40             V1401         2 5d01h         60    360  0  185
1   129.34.9.11             V1510         2 5d01h         73    657  0  440
0   129.34.9.2              V1308         2 5d01h       539   4851  0   23
mmz-b-rtr#
```

EXPLORE ROUTING

HSRP

The core routers, *mmz-x-rtr*, provide Active/Standby high-availability services to *charon-x-vpn*, *ice-x-fw*, and *ga-x-fw* via HSRP. Review the status of the HSRP groups.

show-standby

This script emulates show standby brief. I use it to verify that *mmz-a-rtr* is Active for all interfaces, that *mmz-b-rtr* is Standby for all interfaces, and that the Active/Standby/Group addresses are identified per my expectations.

```
jacobsite> show-standby -e yes -e mmz
```

```
[...]
```

```
For mmz-a-rtr:
```

Int	Grp	Prio	P	D	State	Active Addr	Standby Addr	Group Addr
Vl301	4	105	P	120	active	129.34.0.2	129.34.0.3	129.34.0.1
Vl302	5	105	P	120	active	129.34.0.18	129.34.0.19	129.34.0.17
Vl303	6	105	P	120	active	129.34.8.203	129.34.8.204	129.34.8.201
Vl310	7	105	P	120	active	129.34.0.52	129.34.0.53	129.34.0.51

```
For mmz-b-rtr:
```

Int	Grp	Prio	P	D	State	Active Addr	Standby Addr	Group Addr
Vl301	4	100	P	0	standby	129.34.0.2	129.34.0.3	129.34.0.1
Vl302	5	100	P	0	standby	129.34.0.18	129.34.0.19	129.34.0.17
Vl303	6	100	P	0	standby	129.34.8.203	129.34.8.204	129.34.8.201
Vl310	7	100	P	0	standby	129.34.0.52	129.34.0.53	129.34.0.51

show standby brief

Or, from the CLI:

```
mmz-a-rtr#sh stand br
```

```
          P indicates configured to preempt.
          |
Interface  Grp Prio P State  Active addr  Standby addr  Group addr
Vl301      4   105 P Active  local        129.34.0.3    129.34.0.1
Vl302      5   105 P Active  local        129.34.0.19   129.34.0.17
Vl303      6   105 P Active  local        129.34.8.204  129.34.8.201
Vl310      7   105 P Active  local        129.34.0.53   129.34.0.51
mmz-a-rtr#
```

```
mmz-b-rtr#sh stand br
```

```
          P indicates configured to preempt.
          |
Interface  Grp Prio P State  Active addr  Standby addr  Group addr
Vl301      4   100 P Standby 129.34.0.2    local         129.34.0.1
Vl302      5   100 P Standby 129.34.0.18   local         129.34.0.17
Vl303      6   100 P Standby 129.34.8.203  local         129.34.8.201
Vl310      7   100 P Standby 129.34.0.52   local         129.34.0.51
mmz-b-rtr#
```

Virtualized Route Table

Conceptually, this design divides *mmz-x-rtr* in half, treating the Border (yellow) half as a Core Router handling the BGP reflector function, collecting routing table updates from the Edge Routers (which in turn acquire their tables from their service providers peers). The Border half then routes traffic destined for 129.34.8.201 (the public address of our site-to-site VPN service) across the internal handle-bars and all the rest of the traffic across the external handle-bars.

The Native (orange) half of *mmz-x-rtr* contains the SPA modules, handling the encryption/decryption function of the site-to-site VPN tunnels. It also contains the links to the Firewalls and telecommuter VPN servers demarcating the corporate networks which the MMZ serves.

To deliver virtualized routing¹⁰, this design utilizes Cisco's VRF-Lite function, the same feature which we employ internally to divide our distribution and core layers into Native, Guest, and SCHARP networks.

Notice that this design virtualizes the RIB (Routing Information Base: this is what you see what you type 'show ip route') and the FIB (Forwarding Information Base: what you see when you type 'sh mls cef'), creating two sets of route tables and two sets of forwarding tables. It does not virtualize any other part of the box. For example, VRF-Lite does not virtualize the IEEE802.1q portion of the box (the box contains one set of 4096 802.1q VLANs, all accessible from both routing and forwarding tables. Ditto for other Layer 2 functions, like CDP, and for management functions, like SNMP, syslog and Radius.¹¹

The CLI displays interfaces as belonging to defined VRFs; it does not show interfaces belonging to the Native (aka Default) VRF.

```
mmz-a-rtr#sh ip vrf int
Interface          IP-Address      VRF
Protocol
Vlan306            129.34.8.194   Border          up
Vlan308            129.34.8.3     Border          up
Vlan401            129.34.8.40   Border          up
Vlan410            129.34.8.10   Border          up
Vlan412            129.34.8.12   Border          up
Vlan414            129.34.8.14   Border          up
Vlan416            129.34.8.16   Border          up
Vlan418            129.34.8.18   Border          up
Loopback1          129.34.8.101   Border          up
mmz-a-rtr#
```

```
mmz-b-rtr#sh ip vrf int
```

¹⁰ In order to virtualize a box entirely -- such that CPU, memory, interfaces, and OS are isolated from one another -- one purchases a 'logical router'. Currently, I believe that the only 'logical router' Cisco sells is the CRS-1.

¹¹ Cisco is gradually 'VRFing' management functions. However, as of this writing, the functions listed here are not VRF-aware.

Interface	IP-Address	VRF	
Vlan306	129.34.8.195	Border	up
Vlan308	129.34.9.3	Border	up
Vlan401	129.34.8.41	Border	up
Vlan510	129.34.9.10	Border	up
Vlan512	129.34.9.12	Border	up
Vlan514	129.34.9.14	Border	up
Vlan516	129.34.9.16	Border	up
Vlan518	129.34.9.18	Border	up
Loopback1	129.34.8.102	Border	up

mmz-b-rtr#

Examining the Route Tables

One can of course simply type `sh ip ro` to see the entire route table. If you do not specify a VRF, the IOS will display the route table for the Native VRF. I often find looking at the route table easier if I filter it by source first.

```
sh ip ro static
sh ip ro connected
sh ip ro eigrp
```

Between them, these three commands will show you the entire Native route table. I recommend performing these three commands on each of the MMZ routers, with a copy of the map in front of you. The resulting output will illustrate how each router knows to reach each other, along with principle of "just enough" which I have tried to implement, i.e. each router knows only just enough to reach each other and no more: keep the route tables as small as possible. [We accomplish this through prolific use of 'distribute-list' and 'off-set' list statements inside the 'router eigrp xxx' stanzas.]

On *mmz-x-rtr*, use the above CLI commands to display routes in the Native VRF, and the versions enumerated below to display the routes in the Border VRF.

```
sh ip ro vrf border static
sh ip ro vrf border connected
sh ip ro vrf border eigrp
```

Also, use the following commands to sanity check the physical and logical neighbor arrangements.

```
sh cdp neighbors
sh ip eigrp neighbors
```

and on *mmz-x-rtr* also use:

```
sh ip eigrp vrf border neighbors
```

For BGP the vrf syntax is slightly different:

```
sh ip bgp vpv4 vrf border summary
```

In order to keep the route tables as small as possible, this design employs route filtering between the Border and Native VRFs, leaking only enough of each VRF's routes to keep connectivity functioning.

Walk the Route Tables

MMZ-X-RTR NATIVE VRF

```
show ip route static
```

On *mmz-a-rtr*, the static routes consist of a mix of administratively defined routes ('ip route a.b.c.d e.f.g.h ...') and routes injected by the 'reverse-route' function, part of each crypto map.¹² The routes headed to Null0 are there to efficiently discard frames which we cannot or do not want to route. The static routes for 129.34.240.0/20 and 65.90.32.0/19 send GADGETS traffic to *ga-x-fw*; the static route for 129.34.0.0/16 sends Hutch traffic to *ice-x-fw*.

```
mmz-a-rtr#sh ip ro static
 207.114.139.0/32 is subnetted, 1 subnets
S    206.115.138.10 [1/0] via 129.34.8.194
 216.50.65.0/32 is subnetted, 1 subnets
S    215.61.43.5 [1/0] via 129.34.8.194
 208.51.30.0/32 is subnetted, 1 subnets
S    108.52.41.240 [1/0] via 129.34.8.194
S    169.254.0.0/16 is directly connected, Null0
 116.42.46.0/24 is variably subnetted, 2 subnets, 2 masks
S    116.42.46.0/24 [1/0] via 215.61.43.5
S    116.42.46.3/32 [1/0] via 215.61.43.5
 150.13.0.0/32 is subnetted, 4 subnets
S    150.13.145.142 [1/0] via 120.31.146.2
S    150.13.26.151 [1/0] via 144.122.68.117
S    120.31.146.2 [1/0] via 129.34.8.194
S    144.122.68.117 [1/0] via 129.34.8.194
 203.123.179.0/32 is subnetted, 1 subnets
S    219.125.180.60 [1/0] via 129.34.8.194
 129.34.0.0/16 is variably subnetted, 48 subnets, 8 masks
S    129.34.228.0/23 is directly connected, Null0
S    129.34.226.0/23 is directly connected, Null0
S    129.34.224.0/23 is directly connected, Null0
S    129.34.239.0/24 is directly connected, Null0
S    129.34.238.0/24 is directly connected, Null0
S    129.34.237.0/24 is directly connected, Null0
S    129.34.236.0/24 is directly connected, Null0
S    129.34.235.0/24 is directly connected, Null0
S    129.34.234.0/24 is directly connected, Null0
S    129.34.240.0/20 [1/0] via 129.34.0.20
S    129.34.194.0/23 is directly connected, Null0
S    129.34.192.0/23 is directly connected, Null0
S    129.34.206.0/23 is directly connected, Null0
```

¹² I find this weird, that the 'reverse-route' statement results in a static route in the route table. --sk

```

S      129.34.214.0/23 is directly connected, Null0
S      129.34.208.0/23 is directly connected, Null0
S      129.34.223.0/24 is directly connected, Null0
S      129.34.218.0/23 is directly connected, Null0
S      129.34.216.0/24 is directly connected, Null0
S      129.34.176.0/23 is directly connected, Null0
S      129.34.190.0/24 is directly connected, Null0
S      129.34.135.0/24 is directly connected, Null0
S      129.34.134.0/24 is directly connected, Null0
S      129.34.133.0/24 is directly connected, Null0
S      129.34.132.0/24 is directly connected, Null0
S      129.34.140.0/23 is directly connected, Null0
S      129.34.137.0/24 is directly connected, Null0
S      129.34.136.0/24 is directly connected, Null0
S      129.34.102.0/23 is directly connected, Null0
S      129.34.100.0/23 is directly connected, Null0
S      129.34.96.0/23 is directly connected, Null0
S      129.34.104.0/21 is directly connected, Null0
S      129.34.112.0/21 is directly connected, Null0
S      129.34.86.0/23 is directly connected, Null0
S      129.34.7.0/24 is directly connected, Null0
S      129.34.5.0/24 is directly connected, Null0
S      129.34.0.0/16 [1/0] via 129.34.0.4
S      129.34.8.0/23 is directly connected, Null0
144.68.0.0/25 is subnetted, 1 subnets
S      144.68.126.128 [1/0] via 206.115.138.10
166.94.0.0/16 is variably subnetted, 8 subnets, 2 masks
S      65.94.121.76/32 [1/0] via 129.34.8.194
S      166.94.161.51/32 [1/0] via 65.94.121.76
S      166.94.186.0/24 [1/0] via 127.35.185.14
S      166.94.161.22/32 [1/0] via 65.94.121.76
S      127.35.185.14/32 [1/0] via 129.34.8.194
S      166.94.181.0/24 [1/0] via 128.65.121.225
S      166.94.161.21/32 [1/0] via 65.94.121.76
S      128.65.121.225/32 [1/0] via 129.34.8.194
166.107.0.0/16 is variably subnetted, 2 subnets, 2 masks
S      126.217.166.6/32 [1/0] via 129.34.8.194
S      166.107.169.0/24 [1/0] via 126.217.166.6
212.159.204.0/32 is subnetted, 1 subnets
S      206.160.213.80 [1/0] via 129.34.8.194
S      10.0.0.0/8 is directly connected, Null0
S      152.80.0.0/16 [1/0] via 219.125.180.60
72.0.0.0/19 is subnetted, 1 subnets
S      65.90.32.0 [1/0] via 129.34.0.20
63.0.0.0/32 is subnetted, 1 subnets
S      64.240.36.31 [1/0] via 129.34.8.194
192.168.254.0/32 is subnetted, 2 subnets
S      192.168.254.10 [1/0] via 64.240.36.31
S      192.168.254.11 [1/0] via 64.240.36.31
S      192.0.2.0/24 is directly connected, Null0
S      150.2.0.0/16 [1/0] via 108.52.41.240
S      0.0.0.0/8 is directly connected, Null0
S      172.16.0.0/12 is directly connected, Null0
S      198.18.0.0/15 is directly connected, Null0
S      192.168.0.0/16 is directly connected, Null0
S      192.68.48.0/22 [1/0] via 206.160.213.80

```

show ip route connected

mmz-x-rtr know about these routes because they have interfaces located on these segments.

```

mmz-a-rtr#sh ip ro conn
129.34.0.0/16 is variably subnetted, 48 subnets, 8 masks
C      129.34.8.192/28 is directly connected, Vlan303
C      129.34.8.111/32 is directly connected, Loopback0
InfoTech/Admin/FHCRC
Stuart Kendrick

```

```

C      129.34.0.40/31 is directly connected, Vlan400
C      129.34.0.48/28 is directly connected, Vlan310
C      129.34.8.2/31 is directly connected, Vlan307
C      129.34.0.0/28 is directly connected, Vlan301
C      129.34.0.16/28 is directly connected, Vlan302
mmz-a-rtr#

```

```

mmz-b-rtr#sh ip ro conn
      129.34.0.0/16 is variably subnetted, 48 subnets, 8 masks
C      129.34.8.192/28 is directly connected, Vlan303
C      129.34.0.102/32 is directly connected, Loopback0
C      129.34.0.40/31 is directly connected, Vlan400
C      129.34.0.48/28 is directly connected, Vlan310
C      129.34.9.2/31 is directly connected, Vlan307
C      129.34.0.0/28 is directly connected, Vlan301
C      129.34.0.16/28 is directly connected, Vlan302
mmz-b-rtr#

```

show ip route eigrp

mmz-a-rtr learns only a few routes via EIGRP, notably, the Native VRF Loopback interface from *mmz-b-rtr* (via V1400), the Native-to-Border link on *mmz-b-rtr* (also via V1400), and subnets 1 & 2, via its Border VRF partner. And finally, the gateway-of-last-resort, also learned via its Border VRF partner.

Notice how *mmz-a-rtr* does not see Vlan400 as a path to subnets 1 or 2 ... this because a distribute-list artificially increases the cost of routes learned via V1400, making them unattractive (but available, if *mmz-a-rtr's* Vlan307 path to subnets 1 & 2 goes away).

```

mmz-a-rtr#sh ip ro ei
      129.34.0.0/16 is variably subnetted, 48 subnets, 8 masks
D      129.34.0.102/32 [90/130826] via 129.34.0.41, 03:36:16, Vlan400
D      129.34.9.0/24 [90/3072] via 129.34.8.3, 03:36:12, Vlan307
D      129.34.8.0/24 [90/3072] via 129.34.8.3, 03:36:12, Vlan307
D      129.34.9.2/31 [90/3082] via 129.34.0.41, 03:36:15, Vlan400
D*EX 0.0.0.0/0 [170/63498] via 129.34.8.3, 03:35:50, Vlan307
mmz-a-rtr#

```

In addition to the routes described above for *mmz-a-rtr*, *mmz-b-rtr* also learns many other routes via EIGRP ... because the 'reverse-route' statements on *mmz-a-rtr* inject those routes into *mmz-a-rtr's* table, and *mmz-a-rtr* advertises them to *mmz-b-rtr* via Vlan400, the path which carries EIGRP traffic between *mmz-x-rtr*.

```

mmz-b-rtr#sh ip ro ei
      207.114.139.0/32 is subnetted, 1 subnets
D EX   206.115.138.10 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
      216.50.65.0/32 is subnetted, 1 subnets
D EX   215.61.43.5 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
      208.51.30.0/32 is subnetted, 1 subnets
D EX   108.52.41.240 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
      116.42.46.0/24 is variably subnetted, 2 subnets, 2 masks
D EX   116.42.46.0/24 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
D EX   116.42.46.3/32 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
      150.13.0.0/32 is subnetted, 4 subnets

```

```

D EX 150.13.145.142 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
D EX 150.13.26.151 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
D EX 120.31.146.2 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
D EX 144.122.68.117 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
203.123.179.0/32 is subnetted, 1 subnets
D EX 219.125.180.60 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
129.34.0.0/16 is variably subnetted, 48 subnets, 8 masks
D 129.34.8.111/32 [90/130826] via 129.34.0.40, 05:56:03, Vlan400
D 129.34.8.2/31 [90/3082] via 129.34.0.40, 05:56:03, Vlan400
D 129.34.9.0/24 [90/3072] via 129.34.9.3, 05:23:22, Vlan307
D 129.34.8.0/24 [90/2816] via 129.34.9.3, 05:23:22, Vlan307
144.68.0.0/25 is subnetted, 1 subnets
D EX 144.68.126.128 [170/3082] via 129.34.0.40, 05:52:04, Vlan400
166.94.0.0/16 is variably subnetted, 8 subnets, 2 masks
D EX 65.94.121.76/32 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 166.94.161.51/32 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 166.94.186.0/24 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 166.94.161.22/32 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 127.35.185.14/32 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 166.94.181.0/24 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 166.94.161.21/32 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 128.65.121.225/32 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
166.107.0.0/16 is variably subnetted, 2 subnets, 2 masks
D EX 126.217.166.6/32 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 166.107.169.0/24 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
212.159.204.0/32 is subnetted, 1 subnets
D EX 206.160.213.80 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 152.80.0.0/16 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
63.0.0.0/32 is subnetted, 1 subnets
D EX 64.240.36.31 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
192.168.254.0/32 is subnetted, 2 subnets
D EX 192.168.254.10 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 192.168.254.11 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D EX 150.2.0.0/16 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
D*EX 0.0.0.0/0 [170/63498] via 129.34.9.3, 05:23:28, Vlan307
D EX 192.68.48.0/22 [170/3082] via 129.34.0.40, 05:52:11, Vlan400
mmz-b-rtr#

```

show ip bgp summary

Notice how the Native VRF contains no BGP partners ... because, in fact, the config file defines the BGP process as living in the Border VRF, not in the Native VRF.

```
mmz-a-rtr#sh ip bgp sum
```

```
mmz-a-rtr#
```

```
mmz-b-rtr#sh ip bgp sum
```

```
mmz-b-rtr#
```

MMZ-X-RTR BORDER VRF

Moving to the Border VRF, notice that 'show cdp neighbors' doesn't change -- VRF-Lite virtualizes routing processes, not IEEE functions.

show ip bgp vpnv4 vrf Border summary

The BGP process in the Border VRF 'own's AS 14954; these are the Route Reflectors for our AS. Notice how the syntax has this odd 'vpnv4' phrase in it.

```

mmz-a-rtr#sh ip bgp vpnv4 vrf Border sum
BGP router identifier 129.34.8.111, local AS number 14954
BGP table version is 753522, main routing table version 753522
269813 network entries using 36964381 bytes of memory
877080 path entries using 56133120 bytes of memory
176953/61835 BGP path/bestpath attribute entries using 17695300 bytes of
memory
99293 BGP AS-PATH entries using 2747350 bytes of memory
2029 BGP community entries using 154928 bytes of memory
2 BGP extended community entries using 310 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 113695389 total bytes of memory
BGP activity 271656/1843 prefixes, 891259/14179 paths, scan interval 15 secs

```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	
State/PfxRcd									
129.34.8.102	4	14954	400	423	753522	0	0	06:24:51	2
129.34.8.103	4	14954	35450	425	753522	0	0	06:34:52	170625
129.34.8.104	4	14954	35515	425	753522	0	0	06:34:51	170629
129.34.8.105	4	14954	274453	425	753522	0	0	06:34:26	267911
129.34.8.108	4	14954	274672	424	753522	0	0	06:33:11	267911

```

mmz-b-rtr#sh ip bgp vpnv4 vrf Border sum
BGP router identifier 129.34.0.102, local AS number 14954
BGP table version is 1008690, main routing table version 1008690
269809 network entries using 36963833 bytes of memory
877060 path entries using 56131840 bytes of memory
176964/61848 BGP path/bestpath attribute entries using 17696400 bytes of
memory
99297 BGP AS-PATH entries using 2747446 bytes of memory
2029 BGP community entries using 154928 bytes of memory
2 BGP extended community entries using 310 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 113694757 total bytes of memory
BGP activity 271612/1803 prefixes, 891072/14012 paths, scan interval 15 secs

```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	
State/PfxRcd									
129.34.8.101	4	14954	414	393	1008690	0	0	06:25:36	2
129.34.8.103	4	14954	35375	395	1008690	0	0	06:27:10	170615
129.34.8.104	4	14954	35439	395	1008690	0	0	06:27:10	170619
129.34.8.105	4	14954	272751	393	1008690	0	0	06:25:36	267911
129.34.8.108	4	14954	273112	393	1008690	0	0	06:25:50	267911

show ip route vrf Border connected

The Border VRF sees a list of directly connected (typically point-to-point) routes to the Edge Routers and to the MMZ partner (*mmz-a-rtr* or *mmz-b-rtr*).

```

mmz-a-rtr#sh ip ro vrf Border con
    129.34.0.0/16 is variably subnetted, 33 subnets, 8 masks
C    129.34.8.192/28 is directly connected, Vlan306
C    129.34.8.101/32 is directly connected, Loopback1
C    129.34.8.40/31 is directly connected, Vlan401
C    129.34.8.2/31 is directly connected, Vlan308
C    129.34.8.14/31 is directly connected, Vlan414
C    129.34.8.12/31 is directly connected, Vlan412
C    129.34.8.10/31 is directly connected, Vlan410
C    129.34.8.18/31 is directly connected, Vlan418
C    129.34.8.16/31 is directly connected, Vlan416
mmz-a-rtr#

```

```

mmz-b-rtr#sh ip ro vrf Border con
    129.34.0.0/16 is variably subnetted, 33 subnets, 8 masks
C    129.34.8.192/28 is directly connected, Vlan306
C    129.34.8.102/32 is directly connected, Loopback1
C    129.34.8.40/31 is directly connected, Vlan401
C    129.34.9.2/31 is directly connected, Vlan308
C    129.34.9.12/31 is directly connected, Vlan512
C    129.34.9.14/31 is directly connected, Vlan514
C    129.34.9.10/31 is directly connected, Vlan510
C    129.34.9.16/31 is directly connected, Vlan516
C    129.34.9.18/31 is directly connected, Vlan518
mmz-b-rtr#

```

show ip route vrf Border eigrp

The Border VRF learns about the 'remote' sides of the Edge Routers, those subnets pointing toward the InterNAP and the GigaPOP. In addition, it learns about Loopback interfaces on each of its partners. And finally, it learns about the various routes which the Native VRF advertises to it. A distribute-list blocks the Border VRF from learning any of the 'reverse-route' injected routes from the Native VRF, specifically, the ones injected by the creation of the IPsec tunnels.¹³

In addition, the Border VRF learns how to reach 129.34.0.0/16 and 65.90.32.0/19 via the Vlan308/308 interlink, i.e. via its Native VRF EIGRP partner. This is important – without this advertisement, the Border VRF would not know how to reach the Hutch or the GADGETS, and we would become incredibly secure from Internet attack.

Notice how the gateway-of-last-resort arrives from *internap-x-rtr*.

```

mmz-a-rtr#sh ip ro vrf Border ei
    129.34.0.0/16 is variably subnetted, 33 subnets, 8 masks
D    129.34.8.224/27 [90/28426] via 129.34.8.11, 01:52:18, Vlan410

```

¹³ If this distribute-list went away, then the Border VRF would then that it could reach the remote protected subnets via the Native VRF, and connectivity would break, as mmz-x-rtr would route outbound encrypted frames through the VI305/VI306 interlink ... and the Border VRF would route them back to the Native VRF via the Vlan308/VI307 interlink. Bad.

```

D EX 129.34.240.0/20 [170/3072] via 129.34.8.2, 01:08:39, Vlan308
D 129.34.8.208/28 [90/28426] via 129.34.8.11, 01:52:18, Vlan410
D 129.34.9.128/25 [90/28426] via 129.34.8.11, 01:52:18, Vlan410
D 129.34.8.102/32 [90/130816] via 129.34.8.41, 01:51:54, Vlan401
D 129.34.8.103/32 [90/130826] via 129.34.8.15, 01:51:54, Vlan414
D 129.34.0.102/32 [90/131082] via 129.34.8.2, 01:52:19, Vlan308
D 129.34.8.111/32 [90/130816] via 129.34.8.2, 02:05:38, Vlan308
D 129.34.8.108/32 [90/130826] via 129.34.8.19, 01:51:56, Vlan418
D 129.34.8.104/32 [90/130826] via 129.34.8.17, 01:51:54, Vlan416
D 129.34.8.105/32 [90/130826] via 129.34.8.13, 01:51:55, Vlan412
D 129.34.0.40/31 [90/3072] via 129.34.8.2, 01:52:21, Vlan308
D 129.34.0.48/28 [90/3072] via 129.34.8.2, 02:05:38, Vlan308
D 129.34.9.2/31 [90/3072] via 129.34.8.41, 01:51:54, Vlan401
D 129.34.0.0/28 [90/3072] via 129.34.8.2, 01:52:18, Vlan308
D 129.34.0.0/16 [90/3072] via 129.34.8.2, 01:52:18, Vlan308
D 129.34.9.12/31 [90/3072] via 129.34.8.41, 01:51:52, Vlan401
D 129.34.9.14/31 [90/3072] via 129.34.8.41, 01:51:54, Vlan401
D 129.34.9.10/31 [90/3072] via 129.34.8.41, 01:51:54, Vlan401
D 129.34.9.16/31 [90/3072] via 129.34.8.41, 01:51:54, Vlan401
D 129.34.9.18/31 [90/3072] via 129.34.8.41, 01:51:52, Vlan401
D 129.34.0.16/28 [90/3072] via 129.34.8.2, 02:05:11, Vlan308
66.0.0.0/8 is variably subnetted, 7194 subnets, 14 masks
D 66.150.172.20/30 [90/28426] via 129.34.8.13, 01:51:56, Vlan412
203.123.191.0/31 is subnetted, 2 subnets
D 203.123.191.150 [90/3082] via 129.34.8.15, 01:51:55, Vlan414
D 203.123.191.152 [90/3082] via 129.34.8.17, 01:51:55, Vlan416
203.123.190.0/31 is subnetted, 2 subnets
D 203.123.190.150 [90/3082] via 129.34.8.15, 01:51:55, Vlan414
D 203.123.190.152 [90/3082] via 129.34.8.17, 01:51:55, Vlan416
206.191.144.0/30 is subnetted, 1 subnets
D 206.191.144.136 [90/63242] via 129.34.8.19, 01:51:57, Vlan418
203.123.188.0/31 is subnetted, 2 subnets
D 203.123.188.150 [90/3082] via 129.34.8.15, 01:51:55, Vlan414
D 203.123.188.152 [90/3082] via 129.34.8.17, 01:51:55, Vlan416
72.0.0.0/8 is variably subnetted, 3794 subnets, 15 masks
D EX 65.90.32.0/19 [170/2816] via 129.34.8.2, 01:52:19, Vlan308
63.0.0.0/8 is variably subnetted, 2975 subnets, 16 masks
D 63.251.162.148/30 [90/63242] via 129.34.8.13, 01:51:56, Vlan412
206.253.195.0/24 is variably subnetted, 5 subnets, 4 masks
D 206.253.195.220/30 [90/3082] via 129.34.8.19, 01:51:57, Vlan418
mmz-a-rtr#

```

```

mmz-b-rtr#sh ip ro vrf Border ei
129.34.0.0/16 is variably subnetted, 33 subnets, 8 masks
D 129.34.8.224/27 [90/28426] via 129.34.9.11, 01:52:19, Vlan510
D EX 129.34.240.0/20 [170/3328] via 129.34.8.40, 01:09:05, Vlan401
D 129.34.8.208/28 [90/28426] via 129.34.9.11, 01:52:19, Vlan510
D 129.34.9.128/25 [90/28426] via 129.34.9.11, 01:52:19, Vlan510
D 129.34.8.103/32 [90/130826] via 129.34.9.15, 01:52:19, Vlan514
D 129.34.0.102/32 [90/131338] via 129.34.8.40, 01:52:19, Vlan401
D 129.34.8.111/32 [90/131072] via 129.34.8.40, 01:52:19, Vlan401
D 129.34.8.101/32 [90/130816] via 129.34.8.40, 01:52:19, Vlan401
D 129.34.8.108/32 [90/130826] via 129.34.9.19, 01:52:19, Vlan518

```

```

D      129.34.8.104/32 [90/130826] via 129.34.9.17, 01:52:19, Vlan516
D      129.34.8.105/32 [90/130826] via 129.34.9.13, 01:52:19, Vlan512
D      129.34.0.40/31 [90/3328] via 129.34.8.40, 01:52:19, Vlan401
D      129.34.0.48/28 [90/3328] via 129.34.8.40, 01:52:19, Vlan401
D      129.34.8.2/31 [90/3072] via 129.34.8.40, 01:52:19, Vlan401
D      129.34.0.0/28 [90/3072] via 129.34.9.2, 01:52:45, Vlan308
D      129.34.0.0/16 [90/3072] via 129.34.9.2, 01:52:19, Vlan308
D      129.34.8.14/31 [90/3072] via 129.34.8.40, 01:52:19, Vlan401
D      129.34.8.12/31 [90/3072] via 129.34.8.40, 01:52:19, Vlan401
D      129.34.8.10/31 [90/3072] via 129.34.8.40, 01:52:19, Vlan401
D      129.34.8.18/31 [90/3072] via 129.34.8.40, 01:52:19, Vlan401
D      129.34.8.16/31 [90/3072] via 129.34.8.40, 01:52:19, Vlan401
D      129.34.0.16/28 [90/3328] via 129.34.8.40, 01:52:19, Vlan401
66.0.0.0/8 is variably subnetted, 7194 subnets, 14 masks
D      66.150.172.20/30 [90/28426] via 129.34.9.13, 01:52:20, Vlan512
203.123.191.0/31 is subnetted, 2 subnets
D      203.123.191.150 [90/3082] via 129.34.9.15, 01:52:20, Vlan514
D      203.123.191.152 [90/3082] via 129.34.9.17, 01:52:20, Vlan516
203.123.190.0/31 is subnetted, 2 subnets
D      203.123.190.150 [90/3082] via 129.34.9.15, 01:52:20, Vlan514
D      203.123.190.152 [90/3082] via 129.34.9.17, 01:52:20, Vlan516
206.191.144.0/30 is subnetted, 1 subnets
D      206.191.144.136 [90/63242] via 129.34.9.19, 01:52:20, Vlan518
203.123.188.0/31 is subnetted, 2 subnets
D      203.123.188.150 [90/3082] via 129.34.9.15, 01:52:20, Vlan514
D      203.123.188.152 [90/3082] via 129.34.9.17, 01:52:20, Vlan516
72.0.0.0/8 is variably subnetted, 3794 subnets, 15 masks
D EX   65.90.32.0/19 [170/2816] via 129.34.9.2, 01:52:20, Vlan308
63.0.0.0/8 is variably subnetted, 2975 subnets, 16 masks
D      63.251.162.148/30 [90/63242] via 129.34.9.13, 01:52:20, Vlan512
206.253.195.0/24 is variably subnetted, 5 subnets, 4 masks
D      206.253.195.220/30 [90/3082] via 129.34.9.19, 01:52:20, Vlan518
mmz-b-rtr#

```

show ip route vrf Border static

The Border VRF on *mmz-a-rtr* sees static routes to 129.34.8.0/24 and 129.34.9.0/24 via Null0. If *mmz-a-rtr* cannot find a path to 129.34.8.0/24 or 129.34.9.0/24, then it will efficiently discard traffic destined to those subnets, rather than forwarding them to the Native VRF, which won't know how to deliver them.

```

mmz-a-rtr#sh ip ro vrf Border static
      129.34.0.0/16 is variably subnetted, 33 subnets, 8 masks
S      129.34.9.0/24 is directly connected, Null0
S      129.34.8.0/24 is directly connected, Null0
mmz-a-rtr#

```

```

mmz-b-rtr#sh ip ro vrf Border static
      129.34.0.0/16 is variably subnetted, 33 subnets, 8 masks
S      129.34.9.0/24 is directly connected, Null0
S      129.34.8.0/24 is directly connected, Null0
mmz-b-rtr#

```

INTERNAP-A-RTR

show cdp neighbors

```
internap-a-rtr#sh cdp nei
```

```
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge  
S - Switch, H - Host, I - IGMP, r - Repeater
```

Device ID	Local Intrfce	Holdtme	Capability	Platform	Port ID
mmz-a-rtr.widgets.com	Gig 0/1	137	R S I	WS-C6506	Gig 3/6
mmz-b-rtr.widgets.com	Gig 0/2	163	R S I	WS-C6506	Gig 3/6

```
internap-a-rtr#
```

show ip eigrp neighbors

```
internap-a-rtr#sh ip ei nei
```

```
IP-EIGRP neighbors for process 106
```

H	Address	Interface	Hold Uptime (sec)	SRTT (ms)	RTO	Q	Seq Cnt Num
1	129.34.9.12	Gi0/2	2 03:51:57	1	200	0	48
0	129.34.8.12	Gi0/1	2 04:45:28	1	200	0	2089

```
internap-a-rtr#
```

show ip route connected

```
internap-a-rtr#sh ip ro con
```

```
129.34.0.0/16 is variably subnetted, 30 subnets, 7 masks  
C 129.34.8.105/32 is directly connected, Loopback0  
C 129.34.9.12/31 is directly connected, GigabitEthernet0/2  
C 129.34.8.12/31 is directly connected, GigabitEthernet0/1  
66.0.0.0/8 is variably subnetted, 7067 subnets, 14 masks  
C 66.150.172.20/30 is directly connected, GigabitEthernet0/3  
63.0.0.0/8 is variably subnetted, 2979 subnets, 16 masks  
C 63.251.162.148/30 is directly connected, Serial2/0  
internap-a-rtr#
```

show ip bgp summary

```
internap-a-rtr#sh ip bgp sum
```

```
BGP router identifier 129.34.8.105, local AS number 14954  
BGP table version is 503361, main routing table version 503361  
267574 network entries using 27024974 bytes of memory  
267575 path entries using 12843600 bytes of memory  
75511 BGP path attribute entries using 4562880 bytes of memory  
76024 BGP AS-PATH entries using 2068442 bytes of memory  
6 BGP community entries using 144 bytes of memory  
0 BGP route-map cache entries using 0 bytes of memory  
76046 BGP filter-list cache entries using 912552 bytes of memory  
BGP using 47412592 total bytes of memory  
BGP activity 268606/1032 prefixes, 268705/1130 paths, scan interval 60 secs
```

```

Neighbor      V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
64.252.132.150 4 14744 512159   1731   503361    0    0 04:47:35 267571
129.34.8.101  4 14954   307 331304   503361    0    0 04:47:46    2
129.34.8.102  4 14954   296 449995   503361    0    0 03:53:19    2
internap-a-rtr#

```

show ip route summary

```

internap-a-rtr#sh ip ro sum
IP routing table name is Default-IP-Routing-Table(0)
Route Source      Networks      Subnets      Overhead      Memory (bytes)
connected         0             5             320           760
static            1             0             64            152
eigrp 106         0             36            3520          5472
bgp 14954         130451       137119        17124480      40671660
  External: 267570 Internal: 0 Local: 0
internal          2622
Total             133074       137160        17128384      43751028
internap-a-rtr#

```

INTERNAP-B-RTR

show cdp neighbors

```

internap-b-rtr#sh cdp nei
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce   Holdtme    Capability Platform Port ID
mmz-a-rtr.widgets.com
  Gig 0/1                137        R S I      WS-C6506 Gig 3/8
mmz-b-rtr.widgets.com
  Gig 0/2                162        R S I      WS-C6506 Gig 3/8
internap-b-rtr#

```

show ip eigrp neighbors

```

internap-b-rtr#sh ip eigrp nei
IP-EIGRP neighbors for process 106
H   Address          Interface          Hold Uptime      SRTT   RTO  Q  Seq
                               (sec)            (ms)          Cnt Num
1   129.34.9.18       Gi0/2              2 03:49:45      1    200  0  53
0   129.34.8.18       Gi0/1              2 04:38:09      1    200  0 2088
internap-b-rtr#

```

show ip route connected

```

internap-b-rtr#sh ip ro conn
  129.34.0.0/16 is variably subnetted, 30 subnets, 7 masks
C    129.34.8.108/32 is directly connected, Loopback0
C    129.34.8.18/31 is directly connected, GigabitEthernet0/1
C    129.34.9.18/31 is directly connected, GigabitEthernet0/2
  206.191.144.0/30 is subnetted, 1 subnets
C    206.191.144.136 is directly connected, Serial1/0

```

206.253.195.0/24 is variably subnetted, 6 subnets, 5 masks
C 206.253.195.220/30 is directly connected, GigabitEthernet0/3

show ip bgp summary

```
internap-b-rtr#sh ip bgp sum
BGP router identifier 129.34.8.108, local AS number 14954
BGP table version is 496421, main routing table version 496421
267574 network entries using 27024974 bytes of memory
267575 path entries using 12843600 bytes of memory
75520 BGP path attribute entries using 4531440 bytes of memory
75502 BGP AS-PATH entries using 2054394 bytes of memory
6 BGP community entries using 144 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
50930 BGP filter-list cache entries using 611160 bytes of memory
BGP using 47065712 total bytes of memory
BGP activity 268574/1000 prefixes, 268692/1117 paths, scan interval 60 secs
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
129.34.8.101	4	14954	303	349172	496421	0	0	04:43:32	2
129.34.8.102	4	14954	291	467606	496421	0	0	03:54:09	2
206.191.144.137	4	14744	475238	1705	496421	0	0	04:43:18	267571

internap-b-rtr#

show ip route summary

```
internap-b-rtr#sh ip ro sum
IP routing table name is Default-IP-Routing-Table(0)
Route Source Networks Subnets Overhead Memory (bytes)
connected 0 5 320 760
static 1 0 64 152
eigrp 106 0 36 3520 5472
bgp 14954 130444 137134 17124992 40672876
  External: 267578 Internal: 0 Local: 0
internal 2622 3072984
Total 133067 137175 17128896 43752244
internap-b-rtr#
```

GIGAPOP-A-RTR

show cdp neighbors

```
gigapop-a-rtr>sh cdp nei
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone

Device ID Local Intrfce Holdtme Capability Platform Port ID
mmz-a-rtr.widgets.com
  Gig 1/2 129 R S I WS-C6506 Gig 3/1
mmz-b-rtr.widgets.com
  Gig 1/3 154 R S I WS-C6506 Gig 3/1
gigapop-a-rtr>
```

show ip eigrp neighbors

```
gigapop-a-rtr>sh ip ei nei
```

IP-EIGRP neighbors for process 106

H	Address	Interface	Hold Uptime (sec)	SRTT (ms)	RTO	Q Cnt	Seq Num
1	129.34.9.14	Vl514	2 03:53:40	324	1944	0	49
0	129.34.8.14	Vl414	2 04:35:23	5	200	0	2085

show ip route connected

gigapop-a-rtr>

gigapop-a-rtr>sh ip ro con

129.34.0.0/16 is variably subnetted, 30 subnets, 7 masks
C 129.34.9.14/31 is directly connected, Vlan514
C 129.34.8.14/31 is directly connected, Vlan414
C 129.34.8.103/32 is directly connected, Loopback0
203.123.191.0/31 is subnetted, 2 subnets
C 203.123.191.150 is directly connected, Vlan603
203.123.190.0/31 is subnetted, 2 subnets
C 203.123.190.150 is directly connected, Vlan602
203.123.188.0/31 is subnetted, 2 subnets
C 203.123.188.150 is directly connected, Vlan601
gigapop-a-rtr>

show ip bgp summary

gigapop-a-rtr>sh ip bgp sum

BGP router identifier 129.34.8.103, local AS number 14954
BGP table version is 187986, main routing table version 187986
171192 network entries using 20029464 bytes of memory
179047 path entries using 9310444 bytes of memory
31104/29381 BGP path/bestpath attribute entries using 4354560 bytes of memory
26611 BGP AS-PATH entries using 760482 bytes of memory
2341 BGP community entries using 182564 bytes of memory
5 BGP extended community entries using 572 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
29387 BGP filter-list cache entries using 352644 bytes of memory
BGP using 34990730 total bytes of memory
BGP activity 173341/2147 prefixes, 183174/4127 paths, scan interval 60 secs

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	
129.34.8.101	4	14954	297	34071	187986	0	0	04:37:59	2
129.34.8.102	4	14954	287	63443	187986	0	0	03:56:45	2
203.123.188.150	4	101	40596	1641	187973	0	0	04:38:30	164578
203.123.190.150	4	101	3177	1642	187973	0	0	04:38:36	5077
203.123.191.150	4	101	4526	1641	187973	0	0	04:38:29	9388

gigapop-a-rtr>

show ip route summary

gigapop-a-rtr>sh ip ro sum

IP routing table name is Default-IP-Routing-Table(0)

IP routing table maximum-paths is 32

Route	Source	Networks	Subnets	Overhead	Memory (bytes)
connected		0	6	432	864
static		0	0	0	0
eigrp 106		1	35	3960	5184

```

bgp 14954      83588      87617      12326760    24655564
  External: 171205 Internal: 0 Local: 0
internal      1774
Total         85363      87658      12331152    28543124
Removing Queue Size 0
gigapop-a-rtr>

```

GIGAPOP-B-RTR

show cdp neighbors

```

gigapop-b-rtr>sh cdp nei
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone

```

Device ID	Local Intrfce	Holdtme	Capability	Platform	Port ID
mmz-a-rtr.widgets.com	Gig 1/2	169	R S I	WS-C6506	Gig 3/2
mmz-b-rtr.widgets.com	Gig 1/3	134	R S I	WS-C6506	Gig 3/2

show ip eigrp neighbors

```

gigapop-b-rtr>sh ip ei nei
IP-EIGRP neighbors for process 106
H   Address          Interface          Hold Uptime      SRTT   RTO   Q   Seq
                               (sec)            (ms)            Cnt  Num
1   129.34.9.16       V1516              2 03:57:46      1     200  0   51
0   129.34.8.16       V1416              2 04:31:35      1     200  0  2086

```

show ip ro conn

```

129.34.0.0/16 is variably subnetted, 30 subnets, 7 masks
C    129.34.9.16/31 is directly connected, Vlan516
C    129.34.8.16/31 is directly connected, Vlan416
C    129.34.8.104/32 is directly connected, Loopback0
    203.123.191.0/31 is subnetted, 2 subnets
C    203.123.191.152 is directly connected, Vlan603
    203.123.190.0/31 is subnetted, 2 subnets
C    203.123.190.152 is directly connected, Vlan602
    203.123.188.0/31 is subnetted, 2 subnets
C    203.123.188.152 is directly connected, Vlan601

```

show ip bgp summary

```

gigapop-b-rtr>sh ip bgp sum
BGP router identifier 129.34.8.104, local AS number 14954
BGP table version is 188136, main routing table version 188136
171242 network entries using 20035314 bytes of memory
179096 path entries using 9312992 bytes of memory
31113/29394 BGP path/bestpath attribute entries using 4355820 bytes of memory
26610 BGP AS-PATH entries using 760550 bytes of memory
2341 BGP community entries using 182564 bytes of memory
5 BGP extended community entries using 572 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
29484 BGP filter-list cache entries using 353808 bytes of memory
BGP using 35001620 total bytes of memory

```

BGP activity 173154/1909 prefixes, 183065/3969 paths, scan interval 60 secs

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	
State/PfxRcd									
129.34.8.101	4	14954	293	39167	188136	0	0	04:33:24	2
129.34.8.102	4	14954	282	68532	188136	0	0	03:59:45	2
203.123.188.152	4	101	57108	1615	188136	0	0	04:34:14	164625
203.123.190.152	4	101	4015	1615	188136	0	0	04:34:14	5078
203.123.191.152	4	101	4302	1615	188136	0	0	04:34:15	9389

show ip route summary

```
gigapop-b-rtr>sh ip ro sum
IP routing table name is Default-IP-Routing-Table(0)
IP routing table maximum-paths is 32
Route Source      Networks      Subnets      Overhead      Memory (bytes)
connected         0             6             432           864
static            0             0             0             0
eigrp 106         1             35            3960          5184
bgp 14954         83584        87656         12329280      24660604
  External: 171240 Internal: 0 Local: 0
internal          1774
Total             85359        87697         12333672      28548164
Removing Queue Size 0
gigapop-b-rtr>
```

EXPLORE VLANS

Here, I articulate the function of each of the VLANs. While the VRF function does **not** virtualize VLANs (you only get 4096 802.1q VLANs per device), VLAN interfaces (so long as they are Layer 3 interfaces) get assigned to only one VRF or another. Therefore, I chunk the discussion of each VLAN by the containing VRF.

Overview

For an overview of the defined VLANs and the ports which are employing them, use this command. Notice that this command does not support the 'vrf' syntax: VRF only virtualizes the route table, not the IEEE 802.1q side of the box.

show vlan brief

Notice that V1303, V1400, and V1401 have no physical ports associated with them – looking at the map, notice that these are purely logical interfaces.¹⁴

```
mmz-a-rtr#sh vlan br
```

VLAN Name	Status	Ports
-----------	--------	-------

¹⁴ I quibble here with Cisco's definition of 'logical'; since the IPsec modules show up as Gi5/0/1 and Gi5/0/2 in the config file, I would like to see them show up in this output, too. But the TAC assures me that the fact they don't is by design and reflects the virtual nature of these two ports.

```

-----
1      default                active
301   VLAN0301              active   Fa2/47, Gi3/4
302   VLAN0302              active   Fa2/48, Gi3/5
303   VLAN0303              active
305   VLAN0305              active   Gi4/15
306   VLAN0306              active   Gi4/16
307   VLAN0307              active   Gi4/1
308   VLAN0308              active   Gi4/2
310   VLAN0310              active   Gi4/5
400   VLAN0400              active
401   VLAN0401              active
412   VLAN0412              active   Gi3/6
414   VLAN0414              active   Gi3/1
416   VLAN0416              active   Gi3/2
418   VLAN0418              active   Gi3/8
425   VLAN0425              active   Gi6/1
1002  fddi-default          act/unsup
1003  token-ring-default    act/unsup
1004  fddinet-default       act/unsup
1005  trnet-default         act/unsup
mmz-a-rtr#

```

```
mmz-b-rtr#sh vlan br
```

VLAN	Name	Status	Ports
1	default	active	
301	VLAN0301	active	Fa2/47, Gi3/4
302	VLAN0302	active	Fa2/48, Gi3/5
303	VLAN0303	active	
305	VLAN0305	active	Gi4/15
306	VLAN0306	active	Gi4/16
307	VLAN0307	active	Gi4/1
308	VLAN0308	active	Gi4/2
310	VLAN0310	active	Gi4/5
400	VLAN0400	active	
401	VLAN0401	active	
512	VLAN0512	active	Gi3/6
514	VLAN0514	active	Gi3/1
516	VLAN0516	active	Gi3/2
518	VLAN0518	active	Gi3/8
525	VLAN0525	active	Gi6/1
1002	fddi-default	act/unsup	
1003	token-ring-default	act/unsup	
1004	fddinet-default	act/unsup	
1005	trnet-default	act/unsup	

```
mmz-b-rtr#
```

show interfaces trunk

To see the interfaces which are trunking, use this command. Ignoring the IPsec module for the moment, the only trunked interface in *mmz-x-rtr* is the EtherChannel (Po1) linking them.

The IPsec module ports are a special case – the IOS auto-configures the VLAN parameters around these ports, and while we **can** override the IOS' choices, we run the risk of breaking things if we do. Does it make sense that the Native VLAN for the IPsec ports is 1? No, it does

not – the Native VLAN, to my way of thinking, should be 303. But don't mess with it – the IOS auto-configures these ports, this is what it wants to do, let it do whatever it wants.¹⁵

```
mmz-a-rtr#sh int trunk
```

Port	Mode	Encapsulation	Status	Native vlan
Gi5/0/1	on	802.1q	trunking	1
Gi5/0/2	on	802.1q	trunking	1
Po1	on	802.1q	trunking	301

Port	Vlans allowed on trunk
Gi5/0/1	303
Gi5/0/2	305
Po1	301-302,305,310,400-401

Port	Vlans allowed and active in management domain
Gi5/0/1	303
Gi5/0/2	305
Po1	301-302,305,310,400-401

Port	Vlans in spanning tree forwarding state and not pruned
Gi5/0/1	303
Gi5/0/2	305
Po1	301-302,305,310,400-401

```
mmz-a-rtr#
```

```
mmz-b-rtr#sh int trunk
```

Port	Mode	Encapsulation	Status	Native vlan
Gi5/0/1	on	802.1q	trunking	1
Gi5/0/2	on	802.1q	trunking	1
Po1	on	802.1q	trunking	301

Port	Vlans allowed on trunk
Gi5/0/1	303
Gi5/0/2	305
Po1	301-302,305,310,400-401

Port	Vlans allowed and active in management domain
Gi5/0/1	303
Gi5/0/2	305
Po1	301-302,305,310,400-401

Port	Vlans in spanning tree forwarding state and not pruned
Gi5/0/1	303
Gi5/0/2	305
Po1	301-302,305,310,400-401

```
mmz-b-rtr#
```

¹⁵ This sentence reflects specific advice from a TAC tech wrt configuring the IPSec modules.

Native VRF

Vlan 301

Services *ice-x-fw*. VI301 on *mmz-a-rtr* connects to *ice-a-fw*; VI301 on *mmz-b-rtr* connects to *ice-b-fw*. The EtherChannel linking *mmz-x-rtr* carries the VRRP (*ice-x-fw*) and HSRP (*mmz-x-rtr*) traffic needed to sustain the respective high-availability schemes of these devices.

Vlan 302

Same as VI301, but for the GADGETS.

Vlan 303

Contains the inside interface of the IPsec SPA module. 129.34.8.201 is the virtual address at which our partners point their tunnel terminators. 129.34.8.203 is the real IP address of the IPsec module in *mmz-a-rtr*; 129.34.8.204 is the real IP address of the IPsec module in *mmz-b-rtr*. Through the magic of the IPsec SPA module, VI303 gets bridged across the module to VI305, a Layer 2 only VLAN. These two VLANs carry the HSRP traffic which permits *mmz-x-rtr* to take over from one another as needed. VI305 crosses the EtherChannel between *mmz-x-rtr*. Unencrypted transit traffic arrives on VI303 and gets encrypted as it crosses the IPsec 'bridge' to VI305.

Vlan 305

In-bound, this VLAN carries encrypted traffic, receiving it from VI306 and forwarding it across the IPsec module 'bridge' to VI303 (129.34.8.201). As encrypted traffic traverses the IPsec module, the IPsec module decrypts it. This is the only Layer 2 VLAN in the MMZ. Notice the VLAN mismatch across the encrypted handle-bar link.

Vlan 306

This interface receives encrypted traffic from the Native VRF and forwards that traffic using usual routing rules toward its destination (a tunnel terminator at a partner site). Inbound, the Border VRF carries 129.34.8.192/28 (which contains 129.34.8.201) in its route table, as a directly connected subnet, and thus the Border VRF knows how to forward traffic across the 'encrypted' handle-bar toward VI303. Notice the VLAN mismatch across the encrypted handle-bar link.

Vlan 307

The Native VRF runs EIGRP on this interface and learns how to reach gateway-of-last-resort (as well as subnets 1 & 2) from its partner running on VI308. Notice the VLAN mismatch across the normal handle-bar link. Careful use of 'distribute-list' and 'offset-list' influence the route exchange.

Vlan 308

The Border VRF runs EIGRP on this interface and learns how to reach 65.90.32.0/19 and 129.34.0.0/16 via its EIGRP partner running on VI307. Notice the VLAN mismatch across the normal handle-bar link. Careful use of 'distribute-list' and 'offset-list' influence the route exchange.

Vlan 310

This VLAN services *charon*, the telecommuter VPN server. The EtherChannel carries this VLAN, and the associated VRRP and HSRP traffic which *charon-x-vpn* and *mmz-x-rtr* exchange with each other.

Vlan400

This VLAN carries EIGRP traffic between the Native VRFs within *mmz-x-rtr*. It also carries transit traffic between these two VRFs, under various failure scenarios. Careful use of 'distribute-list' and 'offset-list' influence the route exchange.

Border VRF

Vlan 306

This interface receives encrypted traffic from the Native VRF and forwards that traffic using usual routing rules toward its destination (a tunnel terminator at a partner site). Inbound, the Border VRF carries 129.34.8.192/28 (which contains 129.34.8.201) in its route table, as a directly connected subnet, and thus the Border VRF knows how to forward traffic across the encrypted handle-bar toward V1303. Notice the VLAN mismatch across the encrypted handle-bar link.

Vlan 308

The Border VRF runs EIGRP on this interface and learns how to reach 65.90.32.0/19 and 129.34.0.0/16 via its EIGRP partner running on V1307. Notice the VLAN mismatch across the normal handle-bar link. Careful use of 'distribute-list' and 'offset-list' influence the route exchange.

Vlan401

The Border VRFs exchange routes via EIGRP and BGP across this VLAN; the EtherChannel carries it between *mmz-x-rtr*. This VLAN also carries transit traffic, under various failure scenarios. Careful use of 'distribute-list' and 'offset-list' influence the route exchange.

Vlan 410, 510

This connects to *manwe*, run by WHI. These Vlan interfaces are configured to speak EIGRP with WHI (advertising only the gateway-of-last-resort and accepting advertisements only for the WHI IP space).

Vlan 412, 414, 416, 418

These VLANs link the Edge Routers to the Border VRF in *mmz-a-rtr*. They carry EIGRP and BGP traffic, as well as transit traffic.

Vlan 512, 514, 516, 518

These VLANs link the Edge Routers to the Border VRF in *mmz-b-rtr*. They carry EIGRP and BGP traffic, as well as transit traffic.

HACKS

I'm fond of the two 'handle-bar' paths tying Border VRF to Native VRF. To some people, they may look ugly (particularly if you are sitting in front of the devices physically!). However, to my way of thinking, they provide a drawable way to follow packet flow between these two sides of the device. And once I can draw a path, I can then visualize the packet flows in my head.

However, implementing these paths required implementing two gross hacks: mac-address and Disable CDPv2.

Furthermore, the use of VRF to saw *mmz-x-rtr* in half resulted in the Guest Wireless hack. And finally, the software train which implements Stateful Failover (SSP) for the IPsec modules is not the train which we employ everywhere else on our C6K. In addition, it produces an annoying but spurious log message.

mac-address

First, under the 'interface VLAN30x' definitions, notice the 'mac-address' statements.

```
interface Vlan303
  description IPSEC SPA inside interface Vlan - Layer 3
  mac-address 0016.46b3.2660
interface VLAN305
  description To Border VRF/Encrypted Traffic (IPSEC VPN SPA outside port VLAN - Layer 2)
  mac-address 0003.fe3f.5001
interface VLAN306
  description To Native VRF/Encrypted Traffic
  mac-address 0003.fe3f.5002
interface VLAN307
  description To Border VRF/Normal Traffic
  mac-address 0003.fe3f.5003
interface VLAN308
  description To Native VRF/Normal Traffic
  mac-address 0003.fe3f.5004
```

Why? Well, by default, Catalyst employs the same MAC address on all VLAN interfaces. This sounds like a bad idea, until you remember that, by definition, VLAN interfaces are routed and a router cannot have two interfaces on the same subnet (the IOS will reject efforts to do this). Therefore, a Catalyst will never collide with its own MAC address.

Well, until VRF came around. With VRF, we saw the Catalyst in half ... and now routed VLAN interfaces can share the same subnet. And they do, in the case of subnets 129.34.8.192/28, 129.34.8.2/31, and 129.34.9.2/31.

Personally, I would prefer some command which instructed the Catalyst to use unique MAC addresses on its VLAN interfaces. But I haven't found such a command. Instead, I've found the

'mac-address' command, which allows one to manually set the MAC address. I chose these MAC addresses from a pool of MAC addresses on the old Sup720-3B cards.¹⁶

Disable CDPv2

I like CDP. I like using 'show cdp neighbors' as a quick sanity check on a design. And I like the fact that CDP v2 exchanges duplex information, so that the two devices involved will log duplex mismatch information to syslog (where swatch picks it up and notifies us).

However, CDP v2 also exchanges native VLAN information. And the native VLANs on the interfaces facing each other across the handle-bar paths are different. [Why? Because VRF virtualizes the RIB and FIB but not IEEE 802.1 VLAN numbers.]

So I have disabled CDP v2 on *mmz-x-rtr*. Which means that we are blind to duplex mismatches, as well as the other information with v2 advertises.

```
no cdp advertise-v2
```

Guest Network

Previously, we connected the Guest Network portals/firewalls to *mmz-x-rtr*, where they belong, IMHO, given our use of *mmz-x-rtr* as the Layer 3 core to the MMZ. However, when plugged into *mmz-x-rtr*, these interfaces required policy-based routing in order to push outbound traffic to the InterNAP.

As of this writing, Cisco doesn't support PBR on VRFed interfaces.

So, I moved the portals and plugged them directly into *internap-x-rtr*. This removes the need for policy-based routing statements (yay!) but feels gross to me. And I'm not confident that this approach would support high-availability. [The Guest Network portals aren't highly-available currently, so we're not losing functionality here.]

In the future, I hope to do better. I suspect that the clean way to go is to plug the portals back into *mmz-x-rtr* and to use GRE tunnels to push their traffic to *internap-x-rtr*. But then, if we start using the PNW GigaPOP for Commodity Internet service and drop the InterNAP, as I suspect we'll do, then we'll want to, minimally, borrow Guest Network subnets from the PNW GigaPOP. Or possibly do something else entirely. More thought needed.

Annoying Log Messages

This developers of this rev of the IOS accidentally left a debug 'log' statement in place when they shipped the production release. This log statement produces the following message at regular intervals, visible in syslog:

¹⁶ Might be better to use addresses from one of the current cards in the box; I've struggled with the pros and cons of doing this and haven't found figured out which is best. Statically assigning MAC addresses is gross, no matter how one slices it.

Dec 16 03:28:55 mmz-a-rtr-native 45064: 044951: Dec 16 03:28:56 pst:
IPSEC_FLOW_QUERY KEYS.

and occasionally

Dec 10 04:04:10 mmz-a-rtr-native 465: 000384: Dec 10 04:04:11 pst:
IKEA_SA_QUERY_KEYS.

According to the TAC tech who handled this case, these messages indicate normal behavior of SSP, and the developers have arranged to remove the debug statement in the next production release.

WALK THE CONFIG FILES

In this section, I include copies of the config files, interspersed with comments. I am hoping that as we make change to the production configs that we will remember to update the versions here.

mmz-a-rtr

*! When the booted IOS does not match the version of OS loaded into a SPA module,
! automatically search the file system for suitable fpd packages to flash into the
! SPA modules. We only have one flavor of SPA module (the IPsec one). Here is an example
! of a file system which contains two versions of the IOS and two appropriate versions of the
! IPsec module's flash load.*

```
! mmz-a-rtr#dir disk0:  
! Directory of disk0:/  
!  
!  1  -rw-      80142372  Oct 20 2008 09:13:10 -07:00  s72033 adventerprisek9_wan-mz.122-  
33.SXH3.bin  
!  2  -rw-      80859172  Nov 7 2008 08:31:52 -08:00  s72033-advipservicesk9_wan-mz.122-  
18.SXF15a.bin  
!  3  -rw-      23660032  Oct 20 2008 09:14:44 -07:00  c6500-fpd-pkg.122-33.SXH3.pkg  
!  4  -rw-      20524544  Nov 7 2008 08:36:28 -08:00  c7600-fpd-pkg.122-18.SXF15a.pkg
```

```
! 256540672 bytes total (51343360 bytes free)  
! mmz-a-rtr#
```

*! Once a SPA module has been flashed (once the box has booted once and flashed the SPA
! module, one can delete the fpd package if one wants. We don't do this, however, in order to
! facilitate downgrading to the previous IOS version.*

```
upgrade fpd auto
```

```
!
```

! Inserted by the IOS

```
version 12.2
```

```
no service pad
```

*! Reduces the likelihood that intervening firewalls will drop TCP traffic terminating on
! this device*

```
service tcp-keepalives-in
```

```
service tcp-keepalives-out
```

! Specify format of log messages

```
service timestamps debug datetime localtime show-timezone
```

```

service timestamps log datetime localtime show-timezone
!Encrypt passwords in the config file, to reduce the effect of shoulder surfing
service password-encryption
!Specify format of log messages
service sequence-numbers
!Specify how frequently the management agent updates counters visible from the CLI and SNMP
!interfaces
service counters max age 10
!
hostname mmz-a-rtr
!
boot system flash disk0:s72033-advipservicesk9_wan-mz.122-18.SXF15a.bin
boot system flash disk0:s72033-adventerprisek9_wan-mz.122-33.SXH3.bin
boot system flash sup-bootflash:s72033-boot-mz.122-33.SXH.bin
!Log messages to the internal buffer; this permits the use of the 'show log'
!command and is useful when the device is isolated from its loghosts
logging buffered informational
!Don't send log messages to the console
no logging console
!Don't send log messages to the ssh sessions
no logging monitor
!Define the enable password
enable password 7 secret
!
!Define a local user
username admin password 7 secret
!Enable Radius accounting/authorization/authentication
aaa new-model
aaa authentication login default group radius local
aaa authorization exec default group radius local
aaa accounting exec default start-stop group radius
!
aaa session-id common
clock timezone pst -8
clock summer-time pdt recurring
!When individual channels in an EtherChannel change link status, log that event
logging event link-status default
ip subnet-zero
no ip source-route
!
!
!Enable NetFlow
ip flow-cache timeout active 5
ip tftp source-interface Loopback0
!Don't function as a BOOTP server
no ip bootp server
!Define the 'Border' VRF
ip vrf border
  description Border VRF
  rd 65535:1

```

```

!
! Abandon partially negotiated ssh sessions after 30 seconds
ip ssh time-out 30
! Enable DNS resolution for CLI users
ip domain-name widgets.com
ip name-server 129.34.88.11
ip name-server 129.34.250.12
ip name-server 206.253.194.65
ip accounting-threshold 4000
ipv6 mfib hardware-switching replication-mode ingress
! Enables a protocol (UDLD) which watches for fiber links which are half broken
udld enable
!
udld message time 60
!
! Define a State Synchronization Protocol group ('25' is arbitrary). This protocol will maintain
! (ISAKMP) state between partners sharing the same group number. Specify the HSRP groups
! which SSP will modify (by auto-decrementing their priority) during Stateful Failover events
ssp group 25
  remote 129.34.8.204
  redundancy spa-hsrp
! Castrate VTP
vtp domain WIDGETS
vtp mode transparent
! Define parameters which modify the behavior of the hardware switching functions. I don't
! believe that we have changed the defaults
mls ip slb purge global
mls ip multicast flow-stat-timer 9
mls flow ip interface-destination-source
no mls flow ipv6
mls rate-limit unicast cef receive 10000 100
mls rate-limit unicast ip icmp unreachable acl-drop 0
no mls acl tcam share-global
mls cef error action freeze
!
!
!
!
!
!
!
! Modifies the behavior of redundant Sup cards (we don't have any)
redundancy
  keepalive-enable
  mode sso
  main-cpu
  auto-sync running-config
!
! Modify spanning-tree behavior
spanning-tree mode rapid-pvst
no spanning-tree optimize bpdu transmission

```

```

spanning-tree extend system-id
!
power redundancy-mode combined
! Reduce LACP priority to make mmz-a-rtr the LACP master (with respect to mmz-b-rtr)
lacp system-priority 5
diagnostic cns publish cisco.cns.device.diag_results
diagnostic cns subscribe cisco.cns.device.diag_commands
fabric timer 15
!
! Inserted by the IOS
vlan internal allocation policy ascending
!
! Define the VLANs which we use in this box. Remember to remove VLANs
! from this list when we retire them
vlan 301-303,305-308,310,400-401,412,414,416,418,425
!
!
! Define the policies used during IKE negotiation. I don't understand these stanzas
crypto isakmp policy 5
  encr aes
  authentication pre-share
  group 2
  lifetime 28800
!
crypto isakmp policy 11
  encr 3des
  authentication pre-share
  group 2
  lifetime 28800
!
crypto isakmp policy 21
  encr 3des
  hash md5
  authentication pre-share
  group 2
  lifetime 28800
! Define the shared keys used with our remote partners
crypto isakmp key secret address 108.52.41.240
crypto isakmp key secret address 144.122.68.117
crypto isakmp key secret address 219.125.180.60
crypto isakmp key secret address 120.31.146.2
crypto isakmp key secret address 65.94.121.76
crypto isakmp key secret address 126.217.166.6
crypto isakmp key secret address 206.115.138.10
crypto isakmp key secret address 127.35.185.14
crypto isakmp key secret address 206.160.213.80
crypto isakmp key secret address 215.61.43.5
crypto isakmp key secret address 128.65.121.225
crypto isakmp key secret address 64.240.36.31
! Enable dead peer detection (DPD) with remote partners. This is a Cisco-
! specific keep-alive function; if the remote partner also supports DPD,
! this feature should enable more rapid recovery in the event of failure

```

```

crypto isakmp keepalive 60 5
! Instructs the State Synchronization Protocol group 25 to include isakmp information in the
! state which it maintains
crypto isakmp ssp 25
!
!
! Define the a bunch of encryption behavior which we will associate with each partner. If you
! look closely, you'll see that we employ the same encryption choices with everyone. However,
! separating into tunnel-specific sets supports future differentiation
crypto ipsec transform-set chrncset esp-3des esp-sha-hmac
crypto ipsec transform-set gemsset esp-3des esp-md5-hmac
crypto ipsec transform-set sodset esp-3des esp-md5-hmac
crypto ipsec transform-set mcisset esp-3des esp-md5-hmac
crypto ipsec transform-set pyxisset esp-3des esp-md5-hmac
crypto ipsec transform-set hematopathologysset esp-aes esp-md5-hmac
crypto ipsec transform-set philipsset esp-3des esp-sha-hmac
crypto ipsec transform-set dejarnetteset esp-3des esp-sha-hmac
crypto ipsec transform-set impacset esp-aes esp-sha-hmac
crypto ipsec transform-set icadset esp-3des esp-sha-hmac
!
! Associate the crypto map 'gadgetsvpn' with the local interface Vlan303. This tells the IOS that
! whenever an incoming packet reaches this Vlan interface *and* matches this crypto map, to
! push it through the IPsec module for decryption
crypto map gadgetsvpn local-address Vlan303
! List each of the partners associated with the 'gadgetsvpn' crypto map
crypto map gadgetsvpn 5 ipsec-isakmp
! Define our remote partner
set peer 128.65.121.225
! Renegotiate keys after we have transmitted 4MB
set security-association lifetime kilobytes 4194300
! Use the 'impacset' transform-set (I don't understand this)
set transform-set impacset
! Apply this stanza to traffic matching the 'gadgets-uw-impac' ACL
match address gadgets-uw-impac
! Once this tunnel is up, insert a static route into the local routing table directing traffic destined
! for the subnets specified in 'gadgets-uw-impac' to 129.34.8.194, thus forcing the traffic to cross
! the IPsec module and be encrypted
reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 10 ipsec-isakmp
set peer 126.217.166.6
set transform-set mcisset
match address hutch-mcisprod-mimi3
reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 15 ipsec-isakmp
set peer 65.94.121.76
set transform-set mcisset
match address hutch-mcisprod
reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 25 ipsec-isakmp
set peer 120.31.146.2

```

```

    set transform-set mcisset
    match address hutch-mcis
    reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 30 ipsec-isakmp
    set peer 219.125.180.60
    set transform-set chrncset
    match address hutch-chrnc
    reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 35 ipsec-isakmp
    set peer 144.122.68.117
    set transform-set sodset
    match address hutch-sod
    reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 40 ipsec-isakmp
    set peer 108.52.41.240
    set transform-set gemsset
    match address hutch-gems
    reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 45 ipsec-isakmp
    set peer 206.115.138.10
    set transform-set pyxisset
    match address gadgets-pyxis
    reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 50 ipsec-isakmp
    set peer 127.35.185.14
    set transform-set hematopathologyset
    match address gadgets-hematopathology
    reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 55 ipsec-isakmp
    set peer 206.160.213.80
    set transform-set philipsset
    match address gadgets-philips
    reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 60 ipsec-isakmp
    set peer 215.61.43.5
    set transform-set dejarnetteset
    match address gadgets-dejarnette
    reverse-route remote-peer 129.34.8.194
crypto map gadgetsvpn 65 ipsec-isakmp
    set peer 64.240.36.31
    set transform-set icadset
    match address gadgets-icad
    reverse-route remote-peer 129.34.8.194
!
!
!
!
```

! Define a loopback interface for management functions within the Native VRF. We use this interface as the source/destination for most mmz-a-rtr management traffic (ntp, syslog, Radius, ssh, snmp)

```

interface Loopback0
    ip address 129.34.8.111 255.255.255.255
    no ip redirects
```

```

no ip proxy-arp
!
! Define a loopback interface for management functions within the Native VRF. This allows us
! to anchor the BGP side of the box to the Border VRF
interface Loopback1
 ip vrf forwarding border
 ip address 129.34.8.101 255.255.255.255
 no ip redirects
 no ip proxy-arp
!
! Define the LACP pipe to mmz-b-rtr
interface Port-channel1
 description To mmz-b-rtr
 switchport
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 301
 switchport trunk allowed vlan 301,302,305,310,400,401
 switchport mode trunk
 no ip address
 logging event bundle-status
 no mop enabled
!
interface FastEthernet2/1
 no ip address
 ip flow ingress
 shutdown
!
[... Many unused FastEthernet interfaces here ...]
interface FastEthernet2/46
 no ip address
 ip flow ingress
 shutdown
!
! Useful for those rare occasions when we want to insert a laptop
! into the MMZ
interface FastEthernet2/47
 description Workstation port for WIDGETS Firewall Subnet
 switchport
 switchport access vlan 301
 switchport mode access
 no ip address
 no mop enabled
!
! Useful for those rare occasions when we want to insert a laptop
! into the MMZ
interface FastEthernet2/48
 description Workstation port for GADGETS Firewall Subnet
 switchport
 switchport access vlan 302
 switchport mode access
 no ip address
 no mop enabled
!

```

```

interface GigabitEthernet3/1
  description To gigapop-a-rtr
  switchport
  switchport access vlan 414
  switchport mode access
  no ip address
  no mop enabled
!
interface GigabitEthernet3/2
  description To gigapop-b-rtr
  switchport
  switchport access vlan 416
  switchport mode access
  no ip address
  no mop enabled
!
interface GigabitEthernet3/3
  description To mmz-b-rtr/Channelled Interlink
  switchport
  switchport trunk encapsulation dot1q
  switchport trunk native vlan 301
  switchport trunk allowed vlan 301,302,305,310,401
  switchport mode trunk
  no ip address
  no mop enabled
! Use the LACP protocol for channeling (not the pagp protocol)
channel-protocol lacp
! Put this channel into group 1 and use 'active' LACP mode
channel-group 1 mode active
!
interface GigabitEthernet3/4
  description To Hutch
  switchport
  switchport access vlan 301
  switchport mode access
  no ip address
  no mop enabled
!
interface GigabitEthernet3/5
  description To GADGETS
  switchport
  switchport access vlan 302
  switchport mode access
  no ip address
  no mop enabled
!
interface GigabitEthernet3/6
  description To internap-a-rtr
  switchport
  switchport access vlan 412
  switchport mode access
  no ip address
  no mop enabled

```

```

!
interface GigabitEthernet3/7
  description To mmz-b-rtr/Channelled Interlink
  switchport
  switchport trunk encapsulation dot1q
  switchport trunk native vlan 301
  switchport trunk allowed vlan 301,302,305,310,401
  switchport mode trunk
  no ip address
  no mop enabled
channel-protocol lacp
  channel-group 1 mode active
!
interface GigabitEthernet3/8
  description To internap-b-rtr
  switchport
  switchport access vlan 418
  switchport mode access
  no ip address
  no mop enabled
!
! Build the Normal Handlebar
interface GigabitEthernet4/1
  description To Border VRF/Normal Traffic via Gi4/2
  switchport
  switchport access vlan 307
  switchport mode access
  no ip address
  no mop enabled
!
interface GigabitEthernet4/2
  description To Native VRF/Normal Traffic via Gi4/1
  switchport
  switchport access vlan 308
  switchport mode access
  no ip address
  no mop enabled
!
interface GigabitEthernet4/3
  no ip address
  ip flow ingress
  shutdown
!
interface GigabitEthernet4/4
  no ip address
  ip flow ingress
  shutdown
!
interface GigabitEthernet4/5
  description To charon-a-vpn-outside
  switchport
  switchport access vlan 310
  switchport mode access

```

```

no ip address
no mop enabled
!
interface GigabitEthernet4/6
no ip address
ip flow ingress
shutdown
!
interface GigabitEthernet4/7
no ip address
ip flow ingress
shutdown
!
interface GigabitEthernet4/8
no ip address
ip flow ingress
shutdown
!
interface GigabitEthernet4/9
no ip address
ip flow ingress
shutdown
!
interface GigabitEthernet4/10
no ip address
ip flow ingress
shutdown
!
interface GigabitEthernet4/11
no ip address
ip flow ingress
shutdown
!
interface GigabitEthernet4/12
no ip address
ip flow ingress
shutdown
!
interface GigabitEthernet4/13
no ip address
ip flow ingress
shutdown
!
interface GigabitEthernet4/14
no ip address
ip flow ingress
shutdown
!
! Build the Encrypted Handlebar
interface GigabitEthernet4/15
description To Border VRF/Encrypted Traffic via Gi4/16
switchport
switchport access vlan 305

```

```

switchport mode access
no ip address
!
interface GigabitEthernet4/16
description To Native VRF/Encrypted Traffic via Gi4/15
switchport
switchport access vlan 306
switchport mode access
no ip address
!
interface GigabitEthernet6/1
description To manwe (WHI)
switchport
switchport access vlan 425
switchport mode access
no ip address
!
interface GigabitEthernet6/2
description To Tiki
no ip address
ip flow ingress
no mop enabled
!
interface Vlan1
no ip address
ip flow ingress
!
interface Vlan301
description To Hutch
ip address 129.34.0.2 255.255.255.240
no ip redirects
no ip proxy-arp
ip accounting output-packets
ip flow ingress
! Pass incoming traffic through the 'site-vpn-pbr' route map. If the packet
! matches, do what the route-map sez (forward the frame to 129.34.8.194,
! i.e. across the IPsec module for encryption)
ip policy route-map site-vpn-pbr
no mop enabled
standby 4 ip 129.34.0.1
standby 4 timers 1 3
standby 4 priority 105
standby 4 preempt delay minimum 120
! The 'ssp group 25' command uses 'Widgets-hsrp' to identify an HSRP
! group which it includes in its stateful failover behavior
standby 4 name Widgets-hsrp
!
interface Vlan302
description To GADGETS
ip address 129.34.0.18 255.255.255.240
no ip redirects
no ip proxy-arp
ip accounting output-packets

```

```

ip flow ingress
! Pass incoming traffic through the 'site-vpn-pbr' route map. If the packet
! matches, do what the route-map sez (forward the frame to 129.34.8.194,
! i.e. across the IPsec module for encryption)
ip policy route-map site-vpn-pbr
no mop enabled
standby 5 ip 129.34.0.17
standby 5 timers 1 3
standby 5 priority 105
standby 5 preempt delay minimum 120
! The 'ssp group 25' command uses 'Widgets-hsrp' to identify an HSRP
! group which it includes in its stateful failover behavior
standby 5 name gadgets-hsrp
!
interface Vlan303
description IPSEC SPA inside interface Vlan - Layer 3
! By default, Catalyst uses the same MAC address on all VLAN interfaces. This breaks ARP
! when multiple VLAN interfaces meet across a handlebar path or across an internal path.
! Solve the problem by manually specifying the MAC address for each affected interface. Blech!
mac-address 0016.46b3.2660
ip address 129.34.8.203 255.255.255.240
ip flow ingress
standby 6 ip 129.34.8.201
standby 6 timers 1 3
standby 6 priority 105
standby 6 preempt delay minimum 120
! The 'ssp group 25' command uses 'Widgets-hsrp' to identify an HSRP group which it includes
in
! its stateful failover behavior
standby 6 name spa-hsrp
standby 6 track GigabitEthernet5/0/1 10
standby 6 track GigabitEthernet5/0/2 10
standby 6 track Vlan305 10
standby 6 track Vlan306 10
! Instruct the Stateful Failover group 25 to propagate crypto map gadgetsvpn information
crypto map gadgetsvpn ssp 25
! Employ the IPsec module located in slot 5, subslot 0 for encryption / decryption on this
! interface
crypto engine subslot 5/0
!
interface Vlan305
description To Border VRF/Encrypted Traffic (IPSEC SPA outside port)
mac-address 0003.fe3f.5001
no ip address
ip flow ingress
! Tie this Layer 2 interface to the Layer 3 interface 'VLAN303' for encryption/decryption offload
crypto connect vlan 303
!
interface Vlan306
description To Native VRF/Encrypted Traffic
mac-address 0003.fe3f.5002

```

```

ip vrf forwarding border
ip address 129.34.8.194 255.255.255.240
no ip redirects
no ip proxy-arp
ip accounting output-packets
ip flow ingress
no mop enabled
!
interface Vlan307
description To Border VRF/Normal Traffic
mac-address 0003.fe3f.5003
ip address 129.34.8.2 255.255.255.254
no ip redirects
no ip proxy-arp
ip accounting output-packets
ip flow ingress
ip hello-interval eigrp 106 1
ip hold-time eigrp 106 3
no mop enabled
!
interface Vlan308
description To Native VRF/Normal Traffic
mac-address 0003.fe3f.5004
ip vrf forwarding border
ip address 129.34.8.3 255.255.255.254
no ip redirects
no ip proxy-arp
ip accounting output-packets
ip flow ingress
ip hello-interval eigrp 106 1
ip hold-time eigrp 106 3
no mop enabled
!
interface Vlan310
description To charon-a-vpn-outside
ip address 129.34.0.52 255.255.255.240
no ip redirects
no ip proxy-arp
ip accounting output-packets
ip flow ingress
no mop enabled
standby 7 ip 129.34.0.51
standby 7 timers 1 3
standby 7 priority 105
standby 7 preempt delay minimum 120
standby 7 name charon-hsrp
!
interface Vlan400
description To mmz-b-rtr/Native VRF Interlink
ip address 129.34.0.40 255.255.255.254
no ip redirects
no ip proxy-arp
ip accounting output-packets

```

```

ip flow ingress
ip hello-interval eigrp 106 1
ip hold-time eigrp 106 3
no mop enabled
!
interface Vlan401
description To mmz-b-rtr/Border VRF Interlink
ip vrf forwarding border
ip address 129.34.8.40 255.255.255.254
no ip redirects
no ip proxy-arp
ip accounting output-packets
ip flow ingress
ip hello-interval eigrp 106 1
ip hold-time eigrp 106 3
no mop enabled
!
interface Vlan410
description To manwe (WHI)
ip vrf forwarding border
ip address 129.34.8.25 255.255.255.252
! Filter traffic from WHI, only permitting traffic from the IP spaces defined in the 'whi' ACL
ip access-group whi in
no ip redirects
no ip proxy-arp
ip accounting output-packets
ip flow ingress
ip hello-interval eigrp 106 1
ip hold-time eigrp 106 3
no mop enabled
!
interface Vlan412
description To internap-a-rtr
ip vrf forwarding border
ip address 129.34.8.12 255.255.255.254
no ip redirects
no ip proxy-arp
ip accounting output-packets
ip hello-interval eigrp 106 1
ip hold-time eigrp 106 3
no mop enabled
!
interface Vlan414
description To gigapop-a-rtr
ip vrf forwarding border
ip address 129.34.8.14 255.255.255.254
no ip redirects
no ip proxy-arp
ip accounting output-packets
ip hello-interval eigrp 106 1
ip hold-time eigrp 106 3
no mop enabled
!

```

```

interface Vlan416
  description To gigapop-b-rtr
  ip vrf forwarding border
  ip address 129.34.8.16 255.255.255.254
  no ip redirects
  no ip proxy-arp
  ip accounting output-packets
  ip hello-interval eigrp 106 1
  ip hold-time eigrp 106 3
  no mop enabled
!
interface Vlan418
  description To internap-a-rtr
  ip vrf forwarding border
  ip address 129.34.8.18 255.255.255.254
  no ip redirects
  no ip proxy-arp
  ip accounting output-packets
  ip hello-interval eigrp 106 1
  ip hold-time eigrp 106 3
  no mop enabled
!
router eigrp 106
  ! Tell this EIGRP process to advertise the static routes, but filter that list through the
  ! 'filter-static-native-routes' route-map. The goal is to block the local 'small' routes
  ! (/31, /30, /28) and the local Null0 routes from being propagated to other routers, as
  ! unnecessary additions to their route tables
  redistribute static route-map filter-static-native-routes
  ! Don't bother speaking EIGRP on these interfaces
  passive-interface Vlan301
  passive-interface Vlan302
  passive-interface Vlan303
  passive-interface Vlan310
  passive-interface Loopback0
  ! Artificially increase the administrative distance of routes learned via partners
  ! on Vlan400 by '10'
  offset-list all-routes in 10 Vlan400
  ! Run EIGRP on interfaces which fall within these IP spaces
  network 129.34.0.0 0.0.0.255
  network 129.34.8.2 0.0.0.1
  no auto-summary
  ! When you lose or regain touch with an EIGRP neighbor, log the event
  eigrp log-neighbor-warnings 3600
!
router eigrp 12
  ! Don't bother speaking EIGRP on these interfaces
  passive-interface Vlan306
  passive-interface Loopback1
  ! Artificially increase the administrative distance of routes learned via partners on these VLANs.
  ! These interfaces lead to the edge routers; inserting their local routes into the routing table
  ! gives mmz-a-rtr multiple equal-cost paths to various point-to-point links within the

```

```

! Border VRF. This is unnecessary, as mmz-x-rtr can exchange traffic directly across VLAN401.
! By increasing the administrative distance of routes learned via these interfaces, they are no
! longer equal cost the VLAN401 path is 'shorter'), and they drop out of the routing table.
! If VLAN401 goes away, they will reappear, as being the new cheapest path
offset-list all-routes in 10 Vlan410
offset-list all-routes in 10 Vlan412
offset-list all-routes in 10 Vlan414
offset-list all-routes in 10 Vlan416
offset-list all-routes in 10 Vlan418
no auto-summary
!
! Create an EIGRP process running in the Border VRF
address-family ipv4 vrf border
! Look for static routes flagged as belonging to the Border VRF and advertise them via this
! EIGRP process
redistribute static
! Redistribute gateway-of-last-resort from BGP into EIGRP, for propagation to the Native VRF.
! Without this, the Native VRF will not hear about 0.0.0.0 and thus will not know how to reach
! non-129.34.0.0 addresses. Critical.
redistribute bgp 14954 route-map accept-gateway-of-last-resort
! Run EIGRP on any interfaces falling within these IP spaces
network 129.34.8.0 0.0.0.255
network 129.34.9.0 0.0.0.255
! Only advertise the following routes to the Native VRF: 129.34.8.0/24, 129.34.9.0/24, and
! 0.0.0.0; this to reduce the size of the Native VRF's routing table, making it easier to
! trouble-shoot
distribute-list talk-to-native-vrf out Vlan308
! Only accept the super-routes from the Native VRF, to reduce the size of the Border VRF's
! routing table, making it easier to trouble-shoot
distribute-list listen-to-native-vrf in Vlan308
! Only advertise the gateway-of-last-resort to manwe; no need to fill up its routing table with
! unnecessary gunk
distribute-list talk-to-whi out Vlan410
! Only accept routes defined as belonging to WHI; don't let manwe break stuff by advertising
! reachability to something else, like the gateway-of-last-resort
distribute-list listen-to-whi in Vlan410
no auto-summary
! Advertise this EIGRP process as belonging to group number 106. EIGRP processes will only
! talk to one another if they belong to the same group number. And we want the Native and
! Border VRF EIGRP processes to talk with one another, because we want them to exchange
! routes across the Normal Handlebar. I find this syntax counter-intuitive – it isn't obvious to
! me why Cisco does it this way. How does the IOS know that EIGRP process 12 runs in the
! Native VRF ... and should advertise itself as belonging to process 106, not 12? Well, this is the
! way it handles it
autonomous-system 106
! Specific EIGRP router-id because I'm anal – the default would pick some address which would
! work fine
eigrp router-id 129.34.8.101
eigrp log-neighbor-warnings 3600

```

```

! Leave the Border VRF configuration stanza
exit-address-family
!
! Define the BGP process. 14954 is our ARIN-registered AS number
router bgp 14954
  no synchronization
  bgp cluster-id 10
  bgp log-neighbor-changes
  bgp deterministic-med
! Advertise these super-nets
  no auto-summary
!
! Create a BGP process living inside the Border VRF
  address-family ipv4 vrf border
! Define a bunch of parameters applicable to the 'WIDGETS' group of neighbors
  neighbor WIDGETS peer-group
  neighbor WIDGETS remote-as 14954
  neighbor WIDGETS description iBGP session to mmz-b-rtr
! Locate the source of BGP traffic within the Border VRF
  neighbor WIDGETS update-source Loopback1
  neighbor WIDGETS version 4
  neighbor WIDGETS activate
  neighbor WIDGETS send-community
! Define a bunch of parameters applicable to the 'Border-Router' group of neighbors
  neighbor Border-Router peer-group
  neighbor Border-Router remote-as 14954
  neighbor Border-Router description iBGP/reflector session
! Locate the source of BGP traffic within the Border VRF
  neighbor Border-Router update-source Loopback1
  neighbor Border-Router version 4
  neighbor Border-Router activate
! Send BGP community strings to BGP partners. We don't define any on mmz-x-rtr, so this
! doesn't buy us anything currently
  neighbor Border-Router send-community
! Assign peers to groups
  neighbor 129.34.8.102 peer-group WIDGETS
  neighbor 129.34.8.103 peer-group Border-Router
  neighbor 129.34.8.104 peer-group Border-Router
  neighbor 129.34.8.105 peer-group Border-Router
  neighbor 129.34.8.108 peer-group Border-Router
  no auto-summary
  no synchronization
! Advertise these networks via the BGP process living within the Border VRF
  network 65.90.32.0 mask 255.255.224.0
  network 129.34.0.0
! Leave the Border VRF configuration stanza
exit-address-family
!
! Null routes with no administrative weight are security-related; these are routes for which we
! want to discard traffic and are generally for bogons. Null routes with an administrative weight
! of '240' are routes which we should learn from somewhere else. However, if we aren't learning

```

```

! them, don't just throw up our hands and whine about 'network unreachable'; confidently and
! vigorously discard that traffic by handing it to Null0
ip classless
! We should learn the gateway-of-last-resort via VLAN 307 and the Border VRF. However, if we
! don't, discard traffic bound to unknown destinations
ip route 0.0.0.0 0.0.0.0 Null0 240
! Bogon protection
ip route 0.0.0.0 255.0.0.0 Null0
ip route 10.0.0.0 255.0.0.0 Null0
! Forward GADGETS traffic to ga-x-fw, tag it ('101') for easier handling in route-maps
ip route 65.90.32.0 255.255.224.0 129.34.0.20 tag 101
! If somehow we lose the above static route, discard GADGETS traffic
ip route 65.90.32.0 255.255.224.0 Null0 240
! Bogon protection
ip route 127.0.0.0 255.0.0.0 Null0 240
! Forward GADGETS traffic to ice-x-fw, tag it ('100') for easier handling in route-maps
ip route 129.34.0.0 255.255.0.0 129.34.0.4 tag 100
! If somehow we lose the above static route, discard GADGETS traffic
ip route 129.34.0.0 255.255.0.0 Null0 240
! Discard traffic headed to subnets which we don't currently define. This reduces the load on the
! firewalls when we are under DoS attacks
ip route 129.34.5.0 255.255.255.0 Null0
ip route 129.34.7.0 255.255.255.0 Null0
ip route 129.34.8.0 255.255.254.0 Null0
ip route 129.34.86.0 255.255.254.0 Null0
ip route 129.34.96.0 255.255.254.0 Null0
ip route 129.34.100.0 255.255.254.0 Null0
ip route 129.34.102.0 255.255.254.0 Null0
ip route 129.34.104.0 255.255.248.0 Null0
ip route 129.34.112.0 255.255.248.0 Null0
ip route 129.34.132.0 255.255.255.0 Null0
ip route 129.34.133.0 255.255.255.0 Null0
ip route 129.34.134.0 255.255.255.0 Null0
ip route 129.34.135.0 255.255.255.0 Null0
ip route 129.34.136.0 255.255.255.0 Null0
ip route 129.34.137.0 255.255.255.0 Null0
ip route 129.34.140.0 255.255.254.0 Null0
ip route 129.34.176.0 255.255.254.0 Null0
ip route 129.34.190.0 255.255.255.0 Null0
ip route 129.34.192.0 255.255.254.0 Null0
ip route 129.34.194.0 255.255.254.0 Null0
ip route 129.34.206.0 255.255.254.0 Null0
ip route 129.34.208.0 255.255.254.0 Null0
ip route 129.34.214.0 255.255.254.0 Null0
ip route 129.34.216.0 255.255.255.0 Null0
ip route 129.34.218.0 255.255.254.0 Null0
ip route 129.34.223.0 255.255.255.0 Null0
ip route 129.34.224.0 255.255.254.0 Null0
ip route 129.34.226.0 255.255.254.0 Null0
ip route 129.34.228.0 255.255.254.0 Null0
ip route 129.34.234.0 255.255.255.0 Null0
ip route 129.34.235.0 255.255.255.0 Null0

```

```

ip route 129.34.236.0 255.255.255.0 Null0
ip route 129.34.237.0 255.255.255.0 Null0
ip route 129.34.238.0 255.255.255.0 Null0
ip route 129.34.239.0 255.255.255.0 Null0
! Forward GADGETS traffic to ice-x-fw, tag it ('101') for easier handling in route-maps
ip route 129.34.240.0 255.255.240.0 129.34.0.20 101
! If somehow we lose the above static route, discard GADGETS traffic
ip route 129.34.240.0 255.255.240.0 Null0 240
! Bogon protection
ip route 169.254.0.0 255.255.0.0 Null0 240
ip route 172.16.0.0 255.240.0.0 Null0 240
ip route 192.0.2.0 255.255.255.0 Null0 240
ip route 192.168.0.0 255.255.0.0 Null0 240
ip route 198.18.0.0 255.254.0.0 Null0 240
! These subnets exist only in the Border VRF; if the Border VRF doesn't know how to route a
! packet (via Connected or EIGRP sources), no one does, so discard it. If we don't do this, the
! Native VRF and the Border VRF will forward the frame back and forth (routing loop) until TTL
! on the frame expires and one or the other discards it
ip route vrf border 129.34.8.0 255.255.255.0 Null0
ip route vrf border 129.34.9.0 255.255.255.0 Null0
!
ip bgp-community new-format
! Use the Native VRF's loopback interface as the source for NetFlow reporting
ip flow-export source Loopback0
ip flow-export version 5
no ip http server
!
! Used in route-maps when we want to match all routes
ip access-list standard all-routes
remark *** Multiple Uses: Apply to all routes
permit any
deny any
! Used to restrict access to the management interface (ssh and snmp)
ip access-list standard widgets-and-gadgets
remark *** Security: restrict access to mgmt interfaces
permit 65.90.32.0 0.0.31.255
permit 129.34.0.0 0.0.255.255
deny any
! Used in route-maps to permit passing only 0.0.0.0
ip access-list standard gateway-of-last-resort
remark *** Must: accept the gateway-of-last-resort
permit 0.0.0.0
deny any
! Used in route-maps to accept only routes within our public IP spaces
ip access-list standard listen-to-native-vrf
remark *** Must: accept routes for Hutch & GADGETS IP spaces
permit 65.90.32.0 0.0.31.255
permit 129.34.0.0 0.0.255.255
remark *** Must: discard reverse-route-injected tunnels
deny any
! Used in an interface ACL to prevent WHI from spoofing. Used in route-maps to prevent WHI

```

! from advertising reachability to anything but the IP spaces assigned to them

```
ip access-list standard listen-to-whi
  remark *** Defensive: don't let WHI spoof or advertise broadly
  permit 129.34.8.11
  permit 129.34.9.11
  permit 129.34.8.208 0.0.0.15
  permit 129.34.8.224 0.0.0.31
  permit 129.34.9.128 0.0.0.127
  deny any
```

! Used to restrict access to the SNMP interface

```
ip access-list standard mgmt-stations
  remark *** Security: enumerate the hosts which can write via SNMP
  permit 65.90.34.91
  permit 65.90.34.92
  permit 129.34.66.88
  permit 129.34.34.124
  permit 129.34.80.21
  deny any
```

! Used to limit the routes which the Border VRF advertises to the Native VRF

```
ip access-list standard talk-to-native-vrf
  remark *** Must: propagate gateway-of-last-resort into Native VRF
  permit 0.0.0.0
  remark *** Must: tell the Native VRF how to reach subnets 1 & 2
  permit 129.34.8.0
  permit 129.34.9.0
  deny any
```

! Used to limit the routes which we advertise to WHI (WHI only cares about 0.0.0.0, when talking to us)

```
ip access-list standard talk-to-whi
  remark *** Must: advertise gateway-of-last-resort to WHI
  permit 0.0.0.0
  deny any
```

!

! Define which subnets are protected at the Hutch and which are protected at our partners, where 'protected' means 'we encrypt the traffic and route it via a site-to-site VPN tunnel'. These maps must be identical (and mirror-imaged) on the other side; otherwise, the tunnel won't come up. In the cases where we wanted to protect the entire 129.34.0.0/16 space, we cannot just specify our /16 IP space and be done with it – because the site-to-site VPN tunnel terminator on our end is 129.34.8.201, which falls within the 129.34.0.0/16 space. Technically, as long as we developed an ACL which excluded just this IP address, we'd be fine. But in practice, we have excluded the entire MMZ plus The Pit

!

! Much of the thinking behind these choices is dated, from when we were less certain where the GADGETS would locate its equipment. If we were to revamp these choices today (time-consuming, because it requires coordinating with the tunnel administrators on the far end), we would likely shrink these ACLs and get more precise. For example, in many of these ACLs, we include two subnets from D5SR and J4SR, plus only a few specified addresses within the G Building.

!

! Protect all traffic passing between 129.34.0.0/16 (minus the MMZ and The Pit and

! 152.80.0.0/16. Because SeaChild requires all partner traffic (WIDGETS, GADGETS, and UW) to

! arrive across their GigaPOP connection, this is a requirement

```
ip access-list extended hutch-chrnc
  permit ip 129.34.128.0 0.0.127.255 152.80.0.0 0.0.255.255
  permit ip 129.34.64.0 0.0.63.255 152.80.0.0 0.0.255.255
  permit ip 129.34.32.0 0.0.31.255 152.80.0.0 0.0.255.255
  permit ip 129.34.16.0 0.0.15.255 152.80.0.0 0.0.255.255
  permit ip 129.34.8.0 0.0.7.255 152.80.0.0 0.0.255.255
  permit ip 129.34.4.0 0.0.3.255 152.80.0.0 0.0.255.255
  permit ip 65.90.32.0 0.0.31.255 152.80.0.0 0.0.255.255
  deny ip any any log
```

! Protect all traffic passing between 129.34.0.0/16 (minus the MMZ and The Pit and GE

! Medical Systems (150.2.0.0/16). This is probably overkill, in that GEMS only has gear in the

! GADGETS

```
ip access-list extended hutch-gems
  permit ip 129.34.128.0 0.0.127.255 150.2.0.0 0.0.255.255
  permit ip 129.34.64.0 0.0.63.255 150.2.0.0 0.0.255.255
  permit ip 129.34.32.0 0.0.31.255 150.2.0.0 0.0.255.255
  permit ip 129.34.16.0 0.0.15.255 150.2.0.0 0.0.255.255
  permit ip 129.34.8.0 0.0.7.255 150.2.0.0 0.0.255.255
  permit ip 129.34.4.0 0.0.3.255 150.2.0.0 0.0.255.255
  permit ip 65.90.32.0 0.0.31.255 150.2.0.0 0.0.255.255
  deny ip any any log
```

! Protect two server subnets when talking to this UW host

```
ip access-list extended hutch-mcis
  permit ip 129.34.42.0 0.0.1.255 host 150.13.145.142
  permit ip 129.34.80.0 0.0.1.255 host 150.13.145.142
  deny ip any any log
```

! Protect two server subnets, plus a handful of G Bldg stations, when talking

! to these UW hosts

```
ip access-list extended hutch-mcisprod
  permit ip 129.34.42.0 0.0.1.255 host 166.94.161.51
  permit ip 129.34.80.0 0.0.1.255 host 166.94.161.51
  permit ip host 129.34.245.23 host 166.94.161.51
  permit ip host 129.34.241.74 host 166.94.161.21
  permit ip host 129.34.241.74 host 166.94.161.22
  permit ip host 129.34.241.74 host 166.94.161.51
  permit ip host 65.90.50.169 host 166.94.161.21
  permit ip host 65.90.50.169 host 166.94.161.22
  permit ip host 65.90.50.169 host 166.94.161.51
  deny ip any any log
```

! Protect two server subnets, plus a couple of G Bldg stations, when talking

! to this UW subnet

```
ip access-list extended hutch-mcisprod-mimi3
  permit ip 129.34.42.0 0.0.1.255 166.107.169.0 0.0.0.255
  permit ip 129.34.80.0 0.0.1.255 166.107.169.0 0.0.0.255
  permit ip host 129.34.245.23 166.107.169.0 0.0.0.255
  permit ip host 129.34.241.74 166.107.169.0 0.0.0.255
  permit ip host 65.90.50.168 166.107.169.0 0.0.0.255
  deny ip any any log
```

remark UW hosts are summarized in /24 subnet. Using respective hosts will

break the vpn link with UW's Chec

*! Protect two server subnets, plus a handful of G Bldg stations, when talking
! to a handful of UW hosts*

```
ip access-list extended hutch-site-vpn
permit ip host 129.34.245.15 144.68.126.128 0.0.0.127
permit ip host 129.34.245.23 144.68.126.128 0.0.0.127
permit ip 129.34.42.0 0.0.1.255 host 166.107.169.35
permit ip 129.34.80.0 0.0.1.255 host 166.107.169.35
permit ip 129.34.42.0 0.0.1.255 host 166.107.169.36
permit ip 129.34.80.0 0.0.1.255 host 166.107.169.36
permit ip 129.34.42.0 0.0.1.255 host 166.107.169.37
permit ip 129.34.80.0 0.0.1.255 host 166.107.169.37
permit ip 129.34.42.0 0.0.1.255 host 166.94.161.51
permit ip 129.34.80.0 0.0.1.255 host 166.94.161.51
permit ip 129.34.42.0 0.0.1.255 host 150.13.145.142
permit ip 129.34.80.0 0.0.1.255 host 150.13.145.142
permit ip host 129.34.245.23 host 166.107.169.35
permit ip host 129.34.245.23 host 166.107.169.36
permit ip host 129.34.245.23 host 166.107.169.37
permit ip host 129.34.245.23 host 166.94.161.51
permit ip host 129.34.245.23 host 150.13.145.142
permit ip host 129.34.44.14 host 216.254.22.17
permit ip host 129.34.74.88 host 216.254.22.17
deny ip any any log
```

*! Protect all traffic passing between 129.34.0.0/16 (minus the MMZ and The Pit and this station
! at the UW School of Dentistry.*

```
ip access-list extended hutch-sod
permit ip 129.34.128.0 0.0.127.255 host 150.13.26.151
permit ip 129.34.64.0 0.0.63.255 host 150.13.26.151
permit ip 129.34.32.0 0.0.31.255 host 150.13.26.151
permit ip 129.34.16.0 0.0.15.255 host 150.13.26.151
permit ip 129.34.8.0 0.0.7.255 host 150.13.26.151
permit ip 129.34.4.0 0.0.3.255 host 150.13.26.151
permit ip 65.90.32.0 0.0.31.255 host 150.13.26.151
deny ip any any log
```

*! Protect traffic between a handful of G Bldg stations and
! specified subnets and stations at this vendor's location*

```
ip access-list extended gadgets-dejarnette
permit ip host 65.90.50.48 116.42.46.0 0.0.0.255
permit ip host 65.90.50.49 116.42.46.0 0.0.0.255
permit ip host 65.90.34.91 host 116.42.46.3
permit ip host 65.90.34.92 host 116.42.46.3
deny ip any any log
```

*! Protect traffic from one GB-113 station when talking to this UW
! subnet*

```
ip access-list extended gadgets-hematopathology
permit ip host 129.34.234.91 166.94.186.0 0.0.0.255
permit ip host 65.90.50.37 166.94.186.0 0.0.0.255
deny ip any any log
```

*! Protect traffic from a handful of G Bldg stations and one subnet at
! Phillips*

```
ip access-list extended gadgets-philips
```

```
permit ip host 65.90.42.47 192.68.48.0 0.0.3.255
permit ip host 65.90.34.91 192.68.48.0 0.0.3.255
permit ip host 65.90.34.92 192.68.48.0 0.0.3.255
deny ip any any log
```

*! Protect traffic from two G Bldg stations, when talking to this subnet
! at Pyxis*

```
ip access-list extended gadgets-pyxis
permit ip host 129.34.245.15 144.68.126.128 0.0.0.127
permit ip host 129.34.245.23 144.68.126.128 0.0.0.127
deny ip any any log
```

*! Protect traffic from a handful of GB-113 stations, when talking to this
! UW subnet*

```
ip access-list extended gadgets-uw-impac
deny ip host 65.90.50.91 166.94.181.0 0.0.0.255
deny ip host 65.90.50.92 166.94.181.0 0.0.0.255
permit ip 65.90.50.0 0.0.0.255 166.94.181.0 0.0.0.255
deny ip any any log
```

*! We use this ACL in the policy-based route-map (site-vpn-pbr) which sits on the ingress side of
! the interfaces facing the firewalls. In some sense, this ACL 'sums' all the previous ACLs in this
! section. The site-vpn-pbr instructs mmz-a-rtr to forward all traffic which matches this ACL
! (which is destined for a remote, protected subnet) through the IPsec module for encryption.*

```
ip access-list extended gadgetshutch-site-vpn
remark *** Must: list protected subnets
permit ip 129.34.0.0 0.0.255.255 host 150.13.26.151
permit ip 129.34.0.0 0.0.255.255 152.80.0.0 0.0.255.255
permit ip 129.34.0.0 0.0.255.255 host 152.80.254.10
permit ip host 129.34.245.15 144.68.126.128 0.0.0.127
permit ip host 129.34.245.23 144.68.126.128 0.0.0.127
permit ip 129.34.42.0 0.0.1.255 host 166.107.169.35
permit ip 129.34.80.0 0.0.1.255 host 166.107.169.35
permit ip 129.34.42.0 0.0.1.255 host 166.107.169.36
permit ip 129.34.80.0 0.0.1.255 host 166.107.169.36
permit ip 129.34.42.0 0.0.1.255 host 166.107.169.37
permit ip 129.34.80.0 0.0.1.255 host 166.107.169.37
permit ip 129.34.42.0 0.0.1.255 host 166.94.161.51
permit ip 129.34.80.0 0.0.1.255 host 166.94.161.51
permit ip 129.34.42.0 0.0.1.255 host 150.13.145.142
permit ip 129.34.80.0 0.0.1.255 host 150.13.145.142
permit ip host 129.34.245.23 host 166.107.169.35
permit ip host 129.34.245.23 host 166.107.169.36
permit ip host 129.34.245.23 host 166.107.169.37
permit ip host 129.34.245.23 host 166.94.161.51
permit ip host 129.34.245.23 host 150.13.145.142
permit ip host 129.34.250.36 host 166.94.181.168
permit ip host 129.34.250.37 host 166.94.181.168
permit ip 129.34.42.0 0.0.1.255 host 166.94.181.168
permit ip host 129.34.250.36 host 166.94.181.225
permit ip host 129.34.250.37 host 166.94.181.225
permit ip 129.34.42.0 0.0.1.255 host 166.94.181.225
permit ip 65.90.32.0 0.0.31.255 152.80.0.0 0.0.255.255
permit ip host 129.34.241.74 host 166.107.169.35
permit ip host 129.34.241.74 host 166.107.169.36
```

```
permit ip host 129.34.241.74 host 166.107.169.37
permit ip host 129.34.241.74 host 166.94.161.21
permit ip host 129.34.241.74 host 166.94.161.22
permit ip host 129.34.241.74 host 166.94.161.51
permit ip 65.90.32.0 0.0.31.255 host 150.13.26.151
permit ip 65.90.32.0 0.0.31.255 150.2.0.0 0.0.255.255
permit ip 129.34.128.0 0.0.127.255 150.2.0.0 0.0.255.255
permit ip 129.34.64.0 0.0.63.255 150.2.0.0 0.0.255.255
permit ip 129.34.32.0 0.0.31.255 150.2.0.0 0.0.255.255
permit ip 129.34.16.0 0.0.15.255 150.2.0.0 0.0.255.255
permit ip 129.34.8.0 0.0.7.255 150.2.0.0 0.0.255.255
permit ip 129.34.4.0 0.0.3.255 150.2.0.0 0.0.255.255
permit ip host 65.90.42.47 192.68.48.0 0.0.3.255
permit ip host 65.90.50.48 116.42.46.0 0.0.0.255
permit ip host 65.90.50.49 116.42.46.0 0.0.0.255
permit ip host 65.90.34.91 192.68.48.0 0.0.3.255
permit ip host 65.90.34.92 192.68.48.0 0.0.3.255
permit ip host 65.90.34.91 host 116.42.46.3
permit ip host 65.90.34.92 host 116.42.46.3
```

! These are the Citrix SSL/VPN servers. Why do they need to be excluded from this one tunnel?

```
deny ip host 65.90.50.91 166.94.181.0 0.0.0.255
deny ip host 65.90.50.92 166.94.181.0 0.0.0.255
permit ip 65.90.50.0 0.0.0.255 166.94.181.0 0.0.0.255
permit ip host 65.90.50.169 host 166.94.161.21
permit ip host 65.90.50.169 host 166.94.161.22
permit ip host 65.90.50.169 host 166.94.161.51
permit ip host 65.90.50.168 host 166.107.169.35
permit ip host 65.90.50.168 host 166.107.169.36
permit ip host 65.90.50.168 host 166.107.169.37
permit ip host 65.90.50.37 166.94.186.0 0.0.0.255
permit ip host 129.34.243.44 host 192.168.254.10
permit ip host 129.34.243.44 host 192.168.254.11
permit ip host 129.34.243.48 host 192.168.254.10
permit ip host 129.34.243.48 host 192.168.254.11
permit ip host 65.90.43.48 host 192.168.254.10
permit ip host 65.90.43.48 host 192.168.254.11
permit ip host 65.90.43.49 host 192.168.254.10
permit ip host 65.90.43.49 host 192.168.254.11
permit ip host 65.90.34.91 host 192.168.254.10
permit ip host 65.90.34.91 host 192.168.254.11
```

*! Some boxes at the Hutch produce udp and tcp port 0 packets. I don't know why. At least some
! of these packets don't match any of the above permit lines, so they reach here. Normally, they
! would land on the trailing "deny ip any any log" line. But they are so frequent that I became
! annoyed watching them in syslog when I'm using a 'tail -f syslog | grep 'mmz' to watch the
! results of my work. So I added these lines, to drop them but not to log those events. --sk*

```
remark *** Cosmetic: Block TCP/UDP Port 0 packets; don't log them
deny udp any eq 0 any eq 0
deny tcp any eq 0 any eq 0
remark *** Debug: Log anything that hasn't matched
deny ip any any log
```

!

! Use the loopback0 interface as the source for Radius traffic

```

ip radius source-interface Loopback0
! Save messages of all severity levels (all the way down to debugging) in the on-board log
logging history debugging
! Use Loopback0 as the source for syslog messages
logging source-interface Loopback0
! Send syslog messages to junoite
logging 129.34.44.14
! Send syslog messages to jane
logging 65.90.34.91
! We use this ACL to restrict access to the management interface; useful for places where the
! IOS does not yet support named ACLs
access-list 1 permit 65.90.32.0 0.0.31.255
access-list 1 permit 129.34.0.0 0.0.255.255
access-list 1 deny any
! Restrict which stations can send SNMP Sets and receive copies of the config file via TFTP;
! use in places which do not yet support named ACLs (duplicates the named ACL
! 'widgets-and-gadgets'
access-list 20 permit 65.90.34.91
access-list 20 permit 65.90.34.92
access-list 20 permit 129.34.66.88
access-list 20 permit 129.34.34.124
access-list 20 permit 129.34.80.21
access-list 20 deny any
! Disable CDP v2, which otherwise would whine about the VLAN mismatches between the
! interfaces on either side of each handlebar path. Blech!
no cdp advertise-v2
!
! For traffic which matches the gadgetshutch-site-vpn ACL, forward it to the VLAN306 interface,
! which effectively forces the traffic across the IPsec module, which in turn encrypts it as it flies
route-map site-vpn-pbr permit 10
description *** Must: direct traffic bound for protected subnets thru IPsec
module
match ip address gadgetshutch-site-vpn
set ip next-hop 129.34.8.194
! When redistributing 0.0.0.0 from BGP into EIGRP, we filter the routes using this route-map.
! Without this filter, EIGRP would accept all of BGP's routes, which would populate the EIGRP
! topology map with several hundred thousand routes ... probably a bad idea
route-map accept-gateway-of-last-resort permit 10
description *** Must: redistribute 0.0.0.0 from BGP into EIGRP
match ip address gateway-of-last-resort
!
route-map accept-gateway-of-last-resort deny 20
description *** Must: discard all other routes
match ip address all-routes
!
snmp-server engineID local 000000090200000021000000
! Restrict SNMP activity to the networks and stations defined in ACL 1 and ACL 20
snmp-server community not-secret RO 1
snmp-server community private RW 20
snmp-server community public RO 1
snmp-server packetsize 8192

```

```

snmp-server location "1100 Fairview Ave. N, Room AF-218, Seattle, WA"
snmp-server contact "WIDGETS - Information Technology - Voice/Data
Operations, 206.123.4567, frontdesk@widgets.com"
snmp-server system-shutdown
! Restrict SNMP initiated TFTP file transfers to the stations listed in ACL 20
snmp-server file-transfer access-group 20 protocol tftp
! Restrict the use of SNMP Sets to stations listed in ACL 20
snmp-server tftp-server-list 20
!
! Authenticate SSH and console users via Radius
radius-server host 129.34.52.98 auth-port 1812 acct-port 1813
radius-server host 129.34.252.98 auth-port 1812 acct-port 1813
radius-server source-ports 1645-1646
radius-server retransmit 1
radius-server timeout 3
radius-server key 7 secret
!
control-plane
!
!
! Inserted by the IOS
dial-peer cor custom
!
!
!
banner login _____

```

FRED HUTCHINSON CANCER RESEARCH CENTER

WARNING: To protect this system from unauthorized use and to ensure that this system is functioning properly, activities on this system are monitored and recorded and subject to audit. Use of this system is expressed consent to such monitoring and recording. Any unauthorized access or use of this system is prohibited and could be subject to criminal and civil penalties.

Router: mmz-a-rtr

Location: 1100 Fairview Ave N, AF-218, Seattle, WA
Support: WIDGETS - InfoTech, (206) 123-4567, frontdesk@widgets.com

```

!
line con 0
  exec-timeout 60 0
  logging synchronous level all
line vty 0 4

```

```

session-timeout 60
! Restrict CLI access to stations which reside within our IP spaces
access-class widgets-and-gadgets in
exec-timeout 60 0
logging synchronous level all
transport input ssh
line vty 5 15
session-timeout 60
! Restrict CLI access to stations which reside within our IP spaces
access-class widgets-and-gadgets in
exec-timeout 60 0
logging synchronous level all
transport input ssh
!
!
! Copy traffic traversing the ice-a-fw interface to tiki for packet capture
monitor session 1 source interface Gi3/5
monitor session 1 destination interface Gi6/2
ntp clock-period 17179780
ntp source Loopback0
ntp update-calendar
ntp server 129.34.66.12
ntp server 129.34.98.12
ntp server 129.34.12.12
mac-address-table aging-time 14400
!
end

```

gigapop-a-rtr

Conceptually, *gigapop-x-rtr* are simple: they have two internal interfaces, one each to *mmz-a-rtr* and *mmz-b-rtr*. And one external interface, pointed to the service provider. They exchange routes via eBGP with the service provider. The only complexity relates to how the service provider chunks their L2/L3 connection to us: we employ three VLANs mapped to three separate IP subnets, named (by the service provider) *vrf-com*, *vrf-hp*, and *vrf-nlr*.

vrf-com	Commodity Internet
vrf-hp	High-Performance (Abilence/Internet2)
vrf-nlr	National Lambda Rail

```

upgrade fpd auto
version 12.2
no service pad
service tcp-keepalives-in
service tcp-keepalives-out
service timestamps debug datetime localtime show-timezone
service timestamps log datetime localtime show-timezone
service password-encryption
service sequence-numbers
service counters max age 10
!

```

```

hostname gigapop-a-rtr
!
boot-start-marker
boot system flash disk0:s3223-adventerprisek9_wan-mz.122-33.SXH3.bin
boot system flash disk0:s3223-adventerprisek9_wan-mz.122-33.SXH2a.bin
boot system flash sup-bootdisk:s3223-boot-mz.122-33.SXH.bin
boot-end-marker
!
security authentication failure rate 5 log
security passwords min-length 6
! Track meta information about logging behavior, visible through 'sh log'
logging count
! Track user logins via syslog
logging userinfo
logging buffered informational
no logging console
no logging monitor
enable password 7 secret
!
username admin password 7 secret
aaa new-model
aaa authentication login default group radius local
aaa authorization exec default group radius local
aaa accounting exec default start-stop group radius
!
aaa session-id common
clock timezone pst -8
clock summer-time pdt recurring
logging event link-status default
! Inserted by the IOS; we do not subscribe to this service
call-home
  alert-group configuration
  alert-group diagnostic
  alert-group environment
  alert-group inventory
  alert-group syslog
  profile "CiscoTAC-1"
  no active
  no destination transport-method http
  destination transport-method email
  destination address email callhome@cisco.com
  destination address http
https://tools.cisco.com/its/service/oddce/services/DDCEService
  subscribe-to-alert-group diagnostic severity minor
  subscribe-to-alert-group environment severity minor
  subscribe-to-alert-group syslog severity major pattern ".*"
  subscribe-to-alert-group configuration periodic monthly 20 12:46
  subscribe-to-alert-group inventory periodic monthly 20 12:31
ip subnet-zero
no ip source-route
!
!
!
```

```
ip tftp source-interface Loopback0
no ip bootp server
ip ssh time-out 30
ip domain-name widgets.com
ip name-server 129.34.88.11
ip name-server 129.34.250.12
ip name-server 206.253.194.65
ip accounting-threshold 4000
ipv6 mfib hardware-switching replication-mode ingress
udld enable
```

```
udld message time 60
```

```
vtp domain WIDGETS
vtp mode transparent
no mls acl tcam share-global
mls netflow interface
mls flow ip interface-full
no mls flow ipv6
mls nde sender version 5
mls cef error action freeze
```

```
!
!
!
!
!
!
!
!
```

```
! Log config file changes to syslog
```

```
archive
  log config
  logging enable
  logging size 200
  notify syslog contenttype plaintext
  hidekeys
```

```
!
```

```
redundancy
  keepalive-enable
  mode sso
  main-cpu
  auto-sync running-config
spanning-tree mode pvst
spanning-tree extend system-id
system flowcontrol bus auto
```

```
! I keep forgetting what CNS is ... but we don't use it
```

```
diagnostic cns publish cisco.cns.device.diag_results
diagnostic cns subscribe cisco.cns.device.diag_commands
```

```
!
```

```
vlan internal allocation policy ascending
vlan access-log ratelimit 2000
```

```
!
```

```

vlan 414,514,601-603
!
!
!
!
!
interface Loopback0
 ip address 129.34.8.103 255.255.255.255
 no ip redirects
 no ip proxy-arp
 ip flow ingress
!
interface GigabitEthernet1/1
 description To icar-sttlwa01-02
 switchport
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 601-603
 switchport mode trunk
 no cdp enable
 spanning-tree portfast trunk
!
interface GigabitEthernet1/2
 description To mmz-a-rtr
 switchport
 switchport access vlan 414
 switchport mode access
!
interface GigabitEthernet1/3
 description To mmz-b-rtr
 switchport
 switchport access vlan 514
 switchport mode access
!
interface GigabitEthernet1/4
 no ip address
 ip flow ingress
 shutdown
!
interface GigabitEthernet1/5
 no ip address
 ip flow ingress
 shutdown
!
interface GigabitEthernet1/6
 no ip address
 ip flow ingress
 shutdown
!
interface GigabitEthernet1/7
 no ip address
 ip flow ingress
 shutdown
!

```

```

interface GigabitEthernet1/8
  no ip address
  ip flow ingress
  shutdown
!
interface GigabitEthernet1/9
  no ip address
  ip flow ingress
  shutdown
!
interface Vlan1
  no ip address
  ip flow ingress
  shutdown
!
interface Vlan414
  description To mmz-a-rtr
  ip address 129.34.8.15 255.255.255.254
  no ip redirects
  no ip proxy-arp
  ip accounting output-packets
  ip flow ingress
  ip hello-interval eigrp 106 1
  ip hold-time eigrp 106 3
!
interface Vlan514
  description To mmz-b-rtr
  ip address 129.34.9.15 255.255.255.254
  no ip redirects
  no ip proxy-arp
  ip accounting output-packets
  ip flow ingress
  ip hello-interval eigrp 106 1
  ip hold-time eigrp 106 3
!
interface Vlan601
  description to gigapop vrf-com
  ip address 203.123.188.151 255.255.255.254
  ip access-group edge-security in
  no ip redirects
  no ip proxy-arp
  ip accounting output-packets
  ip accounting access-violations
  ip flow ingress
!
interface Vlan602
  description to gigapop vrf-hp
  ip address 203.123.190.151 255.255.255.254
  ip access-group edge-security in
  no ip redirects
  no ip proxy-arp
  ip accounting output-packets
  ip accounting access-violations

```

```

ip flow ingress
!
interface Vlan603
description to gigapop vrf-nlr
ip address 203.123.191.151 255.255.255.254
ip access-group edge-security in
no ip redirects
no ip proxy-arp
ip accounting output-packets
ip accounting access-violations
ip flow ingress
!
router eigrp 106
passive-interface Vlan601
passive-interface Vlan602
passive-interface Vlan603
network 129.34.8.14 0.0.0.1
network 129.34.8.103 0.0.0.0
network 129.34.9.14 0.0.0.1
network 203.123.188.150 0.0.0.1
network 203.123.190.150 0.0.0.1
network 203.123.191.150 0.0.0.1
eigrp log-neighbor-warnings 3600
!
router bgp 14954
bgp log-neighbor-changes
bgp deterministic-med
neighbor route-reflectors peer-group
neighbor route-reflectors remote-as 14954
neighbor route-reflectors description iBGP/client session to mmz-x-rtr
neighbor route-reflectors update-source Loopback0
neighbor route-reflectors version 4
neighbor eBGP-IPv4-Gigapop peer-group
neighbor eBGP-IPv4-Gigapop remote-as 101
neighbor eBGP-IPv4-Gigapop description IPv4 eBGP session to icar-sttlwa01-02
neighbor eBGP-IPv4-Gigapop password 7 secret
neighbor eBGP-IPv4-Gigapop version 4
neighbor eBGP-IPv4-Gigapop timers 10 30
neighbor 129.34.8.101 remote-as 14954
neighbor 129.34.8.101 peer-group route-reflectors
neighbor 129.34.8.102 remote-as 14954
neighbor 129.34.8.102 peer-group route-reflectors
neighbor 203.123.188.150 remote-as 101
neighbor 203.123.188.150 peer-group eBGP-IPv4-Gigapop
neighbor 203.123.190.150 remote-as 101
neighbor 203.123.190.150 peer-group eBGP-IPv4-Gigapop
neighbor 203.123.191.150 remote-as 101
neighbor 203.123.191.150 peer-group eBGP-IPv4-Gigapop
!
address-family ipv4
  ! Send BGP communities to route-reflectors. Likely unnecessary, as mmz-x-rtr don't examine
  ! communities
  neighbor route-reflectors send-community

```

```

!
! When we receive routes from this peer group, filter the routes through filter-list 51
neighbor route-reflectors filter-list 51 in
neighbor eBGP-IPv4-Gigapop send-community
neighbor eBGP-IPv4-Gigapop route-map eBGP-gp-in in
! When we advertise routes to this peer group, filter the associated AS path through filter-list 51
neighbor eBGP-IPv4-Gigapop filter-list 51 out
neighbor 129.34.8.101 activate
neighbor 129.34.8.102 activate
neighbor 203.123.188.150 activate
neighbor 203.123.190.150 activate
neighbor 203.123.191.150 activate
no auto-summary
no synchronization
exit-address-family
!
ip classless
!
! Define how we handle BGP communities from PNW GigaPOP
ip bgp-community new-format
ip community-list standard gp-client permit 101:20000
ip community-list standard gp-hpeer permit 101:20400
ip community-list standard gp-hnsp permit 101:20200
ip community-list standard gp-cpeer permit 101:20300
ip community-list standard gp-cnsp permit 101:20100
! The point behind this BGP AS-path filter is defensive routing. We do not want to advertise
! routes which contain a non-empty AS path, because, from a BGP point of view, we are an end-
! node,, not a transit provider. We should never advertise reachability to some one else's routes
! (non-empty AS path), only to our own (an empty AS-path)
! When the route contains an empty AS path, permit it
ip as-path access-list 51 permit ^$
! When the route contains a non-empty AS path, deny it
ip as-path access-list 51 deny .*
ip flow-export source Loopback0
ip flow-export version 5
ip flow-export destination 129.34.44.143 2055
!
no ip http server
no ip http secure-server
!
ip access-list standard widgets-and-gadgets
remark *** Security: Restrict access to mgmt interfaces
permit 65.90.32.0 0.0.31.255
permit 129.34.0.0 0.0.255.255
deny any
ip access-list standard mgmt-stations
remark *** Security: enumerate the hosts which can write via SNMP
permit 65.90.34.91
permit 65.90.34.92
permit 129.34.66.88
permit 129.34.34.124
permit 129.34.80.21

```

```

deny any
!
ip access-list extended edge-security
remark *** Block bogons (RFC 1700, 1918, 2544, 3330, 3927)
deny ip 0.0.0.0 0.255.255.255 any
deny ip 10.0.0.0 0.255.255.255 any
deny ip 127.0.0.0 0.255.255.255 any
deny ip 172.16.0.0 0.15.255.255 any
deny ip 169.254.0.0 0.0.255.255 any
deny ip 192.0.2.0 0.0.0.255 any
deny ip 198.18.0.0 0.1.255.255 any
deny ip 192.168.0.0 0.0.255.255 any
deny ip 240.0.0.0 15.255.255.255 any
remark *** Anti-spoofing
deny ip 66.150.172.16 0.0.0.7 any
deny ip 65.90.32.0 0.0.31.255 any
deny ip 129.34.0.0 0.0.255.255 any
deny ip 206.253.195.216 0.0.0.7 any
remark *** Permit WHI
permit ip any 129.34.8.208 0.0.0.15
permit ip any 129.34.8.224 0.0.0.31
permit ip any 129.34.9.128 0.0.0.127
remark *** Permit site-to-site VPN
permit ahp any host 129.34.8.201
permit esp any host 129.34.8.201
permit udp any host 129.34.8.201 eq isakmp
remark *** Permit charon
permit ahp any host 129.34.0.54
permit ahp any host 129.34.0.55
permit esp any host 129.34.0.54
permit esp any host 129.34.0.55
permit tcp any host 129.34.0.54 eq www
permit tcp any host 129.34.0.55 eq www
permit tcp any host 129.34.0.54 eq 443
permit tcp any host 129.34.0.55 eq 443
permit tcp any host 129.34.0.54 eq 10000
permit tcp any host 129.34.0.55 eq 10000
permit udp any host 129.34.0.54 eq 443
permit udp any host 129.34.0.55 eq 443
permit udp any host 129.34.0.54 eq non500-isakmp
permit udp any host 129.34.0.55 eq non500-isakmp
permit udp any host 129.34.0.54 eq isakmp
permit udp any host 129.34.0.55 eq isakmp
remark *** Permit BGP from icar-sttlwa01-02
permit tcp host 203.123.188.150 host 203.123.188.151 eq bgp
permit tcp host 203.123.190.150 host 203.123.190.151 eq bgp
permit tcp host 203.123.191.150 host 203.123.191.151 eq bgp
remark *** Block fragments to MMZ devices
deny tcp any 129.34.0.0 0.0.1.255 fragments
deny tcp any 129.34.9.0 0.0.0.255 fragments
deny udp any 129.34.0.0 0.0.1.255 fragments
deny udp any 129.34.9.0 0.0.0.255 fragments
deny icmp any 129.34.0.0 0.0.1.255 fragments

```

```

deny icmp any 129.34.9.0 0.0.0.255 fragments
remark *** Permit limited ICMP to MMZ devices
permit icmp any 129.34.0.0 0.0.1.255 echo
permit icmp any 129.34.9.0 0.0.0.255 echo
permit icmp any 129.34.0.0 0.0.1.255 echo-reply
permit icmp any 129.34.9.0 0.0.0.255 echo-reply
permit icmp any 129.34.0.0 0.0.1.255 traceroute
permit icmp any 129.34.9.0 0.0.0.255 traceroute
remark *** Deny all other traffic to the MMZ devices
deny ip any 129.34.0.0 0.0.1.255
deny ip any 129.34.9.0 0.0.0.255
remark *** Permit all transit traffic
permit ip any any
remark *** Explicit 'deny all' at the end of an access-list
deny ip any any
!
ip radius source-interface Loopback0
logging history debugging
logging source-interface Loopback0
logging 129.34.44.14
logging 65.90.34.91
access-list 1 permit 65.90.32.0 0.0.31.255
access-list 1 permit 129.34.0.0 0.0.255.255
access-list 1 deny any
!
route-map eBGP-gp-in permit 10
 match community gp-client
!
route-map eBGP-gp-in permit 20
 match community gp-hpeer
!
route-map eBGP-gp-in permit 30
 match community gp-hnsp
!
route-map eBGP-gp-in permit 40
 match community gp-cpeer
!
route-map eBGP-gp-in permit 50
 match community gp-cnsp
!
route-map eBGP-gp-in permit 60
!
snmp-server engineID local 000000090200000021000000
snmp-server community private RW mgmt-stations
snmp-server community public RO widgets-and-gadgets
snmp-server packet-size 8192
snmp-server location "WIDGETS, Phase I, Room AF-218, Seattle, WA"
snmp-server contact "WIDGETS - Information Technology - Voice/Data
Operations, 206.123.4567, frontdesk@widgets.com"
snmp-server system-shutdown
snmp-server file-transfer access-group mgmt-stations protocol tftp
snmp-server tftp-server-list mgmt-stations
!

```

```
radius-server host 129.34.52.98 auth-port 1812 acct-port 1813
radius-server host 129.34.252.98 auth-port 1812 acct-port 1813
radius-server source-ports 1645-1646
radius-server retransmit 1
radius-server timeout 3
radius-server key 7 secret
!
control-plane
!
!
! Inserted by the IOS; we don't use this
dial-peer cor custom
!
!
!
banner login _____
```

FRED HUTCHINSON CANCER RESEARCH CENTER

WARNING: To protect this system from unauthorized use and to ensure that this system is functioning properly, activities on this system are monitored and recorded and subject to audit. Use of this system is expressed consent to such monitoring and recording. Any unauthorized access or use of this system is prohibited and could be subject to criminal and civil penalties.

Router: gigapop-a-rtr

Location: 1100 Fairview Ave N, AF-218, Seattle, WA

Support: WIDGETS - InfoTech, (206) 123-4567, frontdesk@widgets.com

```
!
line con 0
  exec-timeout 60 0
  logging synchronous level all
line vty 0 4
  session-timeout 60
  access-class widgets-and-gadgets in
  exec-timeout 60 0
  password 7 022A0B5400400B33434125
  logging synchronous level all
  transport input ssh
line vty 5 15
  session-timeout 60
  access-class widgets-and-gadgets in
```

```

exec-timeout 60 0
password 7 022A0B5400400B33434125
logging synchronous level all
transport input ssh
!
ntp clock-period 17179857
ntp source Loopback0
ntp update-calendar
ntp server 129.34.66.12
ntp server 129.34.98.12
ntp server 129.34.12.12
mac-address-table aging-time 14400
!
end

```

internap-a-rtr

Conceptually, *internap-x-rtr* are simple: they have two internal interfaces, one each to *mmz-a-rtr* and *mmz-b-rtr*. And one external interface, pointed to the service provider. They exchange routes via eBGP with the service provider.

In addition, the two Guest Network firewalls/portals attach here; *cf-ptl* to *internap-a-rtr* and *md-ptl* to *internap-b-rtr*. We have rented two /29 networks from the InterNAP, with the intent of putting our Guest Network traffic onto these ranges, and *internap-x-rtr* both contain null routes, with high administrative weights, to sink traffic destined to these spaces. Currently, we only deploy two /30s from these ranges (one to *cf-ptl* and the other to *md-ptl*).

```

version 12.3
no service pad
service tcp-keepalives-in
service tcp-keepalives-out
service timestamps debug datetime localtime show-timezone
service timestamps log datetime localtime show-timezone
service password-encryption
service sequence-numbers
!
hostname internap-a-rtr
!
boot-start-marker
boot system flash disk2:c7200-jk9s-mz.123-26.bin
boot system flash disk2:c7200-jk9s-mz.123-24.bin
boot system flash bootflash:c7200-kboot-mz.123-26.bin
boot bootldr bootflash:c7200-kboot-mz.123-26.bin
boot-end-marker
!
security authentication failure rate 5 log
security passwords min-length 6
logging queue-limit 100
logging count
logging buffered informational

```



```

interface GigabitEthernet0/1
  description To mmz-a-rtr
  ip address 129.34.8.13 255.255.255.254
  no ip redirects
  no ip proxy-arp
  ip hello-interval eigrp 106 1
  ip hold-time eigrp 106 3
  ip route-cache flow
  duplex auto
  speed auto
  media-type gbic
  negotiation auto
!
interface GigabitEthernet0/2
  description To mmz-b-rtr
  ip address 129.34.9.13 255.255.255.254
  no ip redirects
  no ip proxy-arp
  ip hello-interval eigrp 106 1
  ip hold-time eigrp 106 3
  ip route-cache flow
  duplex auto
  speed auto
  media-type gbic
  negotiation auto
!
interface GigabitEthernet0/3
  description To cf-pt1
  ip address 66.150.172.21 255.255.255.252
  ip access-group edge-security in
  no ip redirects
  no ip proxy-arp
  duplex auto
  ip route-cache flow
  speed auto
  media-type rj45
  negotiation auto
  no cdp enable
!
interface FastEthernet1/0
  no ip address
  shutdown
!
interface Serial2/0
  description To border6
  ip address 63.251.162.150 255.255.255.252
  ip access-group edge-security in
  no ip redirects
  no ip proxy-arp
  ip accounting output-packets
  ip accounting access-violations
  ip route-cache flow
  dsu mode 1

```

```

dsu bandwidth 44210
framing c-bit
cablelength 60
serial restart-delay 0
fair-queue
no cdp enable
!
router eigrp 106
 redistribute static
 passive-interface GigabitEthernet0/3
 passive-interface Serial2/0
 network 63.251.162.148 0.0.0.3
 network 66.150.172.21 0.0.0.3
 network 129.34.8.12 0.0.0.1
 network 129.34.8.105 0.0.0.0
 network 129.34.9.12 0.0.0.1
 no auto-summary
 eigrp log-neighbor-warnings 3600
!
router bgp 14954
 no synchronization
 bgp log-neighbor-changes
 bgp deterministic-med
 neighbor route-reflectors peer-group
 neighbor route-reflectors remote-as 14954
 neighbor route-reflectors description iBGP/client session to mmz-x-rtr
 neighbor route-reflectors update-source Loopback0
 neighbor route-reflectors send-community
 neighbor route-reflectors version 4
 neighbor route-reflectors filter-list 51 in
 neighbor fisher-plaza peer-group
 neighbor fisher-plaza remote-as 14744
 neighbor fisher-plaza description eBGP session to border4
 neighbor fisher-plaza version 4
 neighbor fisher-plaza timers 10 30
 neighbor fisher-plaza password 7 secret
 neighbor fisher-plaza filter-list 51 out
 neighbor 64.252.132.150 peer-group fisher-plaza
 neighbor 129.34.8.101 peer-group route-reflectors
 neighbor 129.34.8.102 peer-group route-reflectors
 no auto-summary
!
ip classless
ip route 66.150.172.16 0.0.0.7 Null0 200
ip flow-export source Loopback0
ip flow-export version 5
ip flow-export destination 129.34.44.143 2055
no ip http server
no ip http secure-server
!
ip as-path access-list 51 permit ^$
ip as-path access-list 51 deny .*
!

```

```

!
ip access-list standard widgets-and-gadgets
  remark *** Security: Restrict access to mgmt interfaces
  permit 65.90.32.0 0.0.31.255
  permit 129.34.0.0 0.0.255.255
  deny any
ip access-list standard mgmt-stations
  remark *** Security: enumerate the hosts which can write via SNMP
  permit 65.90.34.91
  permit 65.90.34.92
  permit 129.34.66.88
  permit 129.34.34.124
  permit 129.34.80.21
  deny any
!
ip access-list extended edge-security
  remark *** Block Microsoft traffic
  deny tcp any 129.34.0.0 0.0.255.255 range 135 139
  deny udp any 129.34.0.0 0.0.255.255 range 135 netbios-ss
  deny tcp any 129.34.0.0 0.0.255.255 eq 445
  deny tcp any 129.34.0.0 0.0.255.255 eq 1434
  remark *** Block bogons (RFC 1700, 1918, 2544, 3330, 3927)
  deny ip 0.0.0.0 0.255.255.255 any
  deny ip 10.0.0.0 0.255.255.255 any
  deny ip 127.0.0.0 0.255.255.255 any
  deny ip 172.16.0.0 0.15.255.255 any
  deny ip 169.254.0.0 0.0.255.255 any
  deny ip 192.0.2.0 0.0.0.255 any
  deny ip 198.18.0.0 0.1.255.255 any
  deny ip 192.168.0.0 0.0.255.255 any
  deny ip 240.0.0.0 15.255.255.255 any
  ! Notice that we do not protect against Guest Network spoofing
  remark *** Anti-spoofing
  deny ip 65.90.32.0 0.0.31.255 any
  deny ip 129.34.0.0 0.0.255.255 any
  remark *** Permit WHI
  permit ip any 129.34.8.208 0.0.0.15
  permit ip any 129.34.8.224 0.0.0.31
  permit ip any 129.34.9.128 0.0.0.127
  remark *** Permit site-to-site VPN
  permit ahp any host 129.34.8.201
  permit esp any host 129.34.8.201
  permit udp any host 129.34.8.201 eq isakmp
  remark *** Permit charon
  permit ahp any host 129.34.0.54
  permit ahp any host 129.34.0.55
  permit esp any host 129.34.0.54
  permit esp any host 129.34.0.55
  permit tcp any host 129.34.0.54 eq MM
  permit tcp any host 129.34.0.55 eq MM
  permit tcp any host 129.34.0.54 eq 443
  permit tcp any host 129.34.0.55 eq 443
  permit tcp any host 129.34.0.54 eq 10000

```

```

permit tcp any host 129.34.0.55 eq 10000
permit udp any host 129.34.0.54 eq 443
permit udp any host 129.34.0.55 eq 443
permit udp any host 129.34.0.54 eq non500-isakmp
permit udp any host 129.34.0.55 eq non500-isakmp
permit udp any host 129.34.0.54 eq isakmp
permit udp any host 129.34.0.55 eq isakmp
remark *** Permit BGP from border6.s7-11.Widgets-4.sef.pnap.net
permit tcp host 64.252.132.150 host 63.215.162.150 eq bgp
remark *** Block fragments to MMZ devices
deny tcp any 129.34.0.0 0.0.1.255 fragments
deny tcp any 129.34.9.0 0.0.0.255 fragments
deny udp any 129.34.0.0 0.0.1.255 fragments
deny udp any 129.34.9.0 0.0.0.255 fragments
deny icmp any 129.34.0.0 0.0.1.255 fragments
deny icmp any 129.34.9.0 0.0.0.255 fragments
remark *** Permit limited ICMP to MMZ devices
permit icmp any 129.34.0.0 0.0.1.255 echo
permit icmp any 129.34.9.0 0.0.0.255 echo
permit icmp any 129.34.0.0 0.0.1.255 echo-reply
permit icmp any 129.34.9.0 0.0.0.255 echo-reply
permit icmp any 129.34.0.0 0.0.1.255 traceroute
permit icmp any 129.34.9.0 0.0.0.255 traceroute
remark *** Deny all other traffic to the MMZ devices
deny ip any 129.34.0.0 0.0.1.255
deny ip any 129.34.9.0 0.0.0.255
remark *** Permit all transit traffic
permit ip any any
remark *** Explicit 'deny all' at the end of an access-list
deny ip any any
ip radius source-interface Loopback0
logging history debugging
logging source-interface Loopback0
logging 129.34.44.14
logging 65.90.34.91
access-list 1 permit 129.34.0.0 0.0.255.255
access-list 1 permit 65.90.32.0 0.0.31.255
access-list 1 deny any log
access-list 20 permit 65.90.34.91
access-list 20 permit 65.90.34.92
access-list 20 permit 129.34.66.88
access-list 20 permit 129.34.34.124
access-list 20 permit 129.34.80.21
access-list 20 deny any
!
! Prepend our AS number to our route advertisements; this artificially increases the AS hop
! count to our networks via the InterNAP and, we hope, increases the chances that remote
! routers will push traffic destined to us across the GigaPOP (cheaper) rather than across the
! InterNAP
route-map eBGP-transit-AS14744-out permit 10
set as-path prepend 14954 14954
!
! Accept the gateway-of-last-resort advertisement from the InterNAP and propagate it into

```

! EIGRP ... but don't propagate any of the other BGP-derived routes!

```
route-map accept-only-gateway-of-last-resort permit 10
  description *** Must: redistribute 0.0.0.0 from BGP into EIGRP
  match ip address gateway-of-last-resort
```

!

```
route-map accept-only-gateway-of-last-resort deny 20
```

```
  description *** Must: discard all other routes
  match ip address all-routes
```

!

*! This version of the IOS does not support named access-lists for SNMP, ergo the use of
! numbered access lists: ACL 20 and ACL 1*

```
snmp-server engineID local 000000090200000021000000
snmp-server community private RW 20
snmp-server community public RO 1
snmp-server packetsize 8192
snmp-server location "1100 Fairview Ave. N., Room CF114, Seattle, WA"
snmp-server contact "WIDGETS - CNS, (206) 123-4567, frontdesk@widgets.com"
snmp-server system-shutdown
snmp-server enable traps tty
snmp-server tftp-server-list 20
```

!

!

```
radius-server host 129.34.52.98 auth-port 1812 acct-port 1813
radius-server host 129.34.252.98 auth-port 1812 acct-port 1813
radius-server retransmit 1
radius-server timeout 3
radius-server key 7 secret
```

!

!

! Inserted by the IOS; we don't use this

```
dial-peer cor custom
```

!

!

!

!

! Inserted by the IOS; we don't use this

```
gatekeeper
  shutdown
```

!

```
banner login _____
```

FRED HUTCHINSON CANCER RESEARCH CENTER

WARNING: To protect this system from unauthorized use and to ensure that this system is functioning properly, activities on this system are monitored and recorded and subject to audit. Use of this system is expressed consent to such monitoring and recording. Any unauthorized access or use of this system is prohibited and could be subject to criminal and civil penalties.

Router: internap-a-rtr

Location: 1100 Fairview Ave N, AF-218, Seattle, WA

Support: WIDGETS - InfoTech, (206) 123-4567, frontdesk@widgets.com

```
!  
line con 0  
  exec-timeout 60 0  
  logging synchronous level all  
  stopbits 1  
line aux 0  
  logging synchronous level all  
  stopbits 1  
line vty 0 4  
  session-timeout 60  
  access-class widgets-and-gadgets in  
  exec-timeout 0 0  
  password 7 secret  
  logging synchronous level all  
  transport input ssh  
line vty 5 15  
  session-timeout 60  
  access-class widgets-and-gadgets in  
  exec-timeout 0 0  
  password 7 secret  
  logging synchronous level all  
  transport input ssh  
!  
ntp clock-period 17179946  
ntp source Loopback0  
ntp update-calendar  
ntp server 129.34.66.12  
ntp server 129.34.98.12  
ntp server 129.34.12.12  
!  
end
```