Troubleshooting Cisco N9300 Smart Switches with Hypershield

DPU Bringup

Successful DPU bringup is required for the Smart Switch to provide security functions. These commands check the status of the DPUs after enabling the service acceleration feature.

Procedure

Step 1 Verify that the **service-acceleration** feature is enabled.

Example:

```
switch# show run service-acceleration | grep feature
feature service-acceleration
```

Step 2 (Optional) Verify the feature status using the **show feature** command.

Example:

```
show feature | grep service-acceleration
service-acceleration   1          enabled
```

Step 3 Verify that the DPUs are powered up and online.

Example:

```
switch# show module | begin DPU
Mod DPU   Module-Type           Model           Status
--- ---  -------------- --------------------- --------
1    1     DPU             N9324C-SE1U-DPU       ok
1    2     DPU             N9324C-SE1U-DPU       ok
1    3     DPU             N9324C-SE1U-DPU       ok
1    4     DPU             N9324C-SE1U-DPU       ok
```

Ensure that all listed DPUs **Status** is **ok**.

Step 4 Verify the DPU software and hardware versions.

Example:

```
switch# show module | begin DPU
 Mod DPU     Sw             Hw      Serial-Num   Online Diag Status
 --- ---  -------------- ---------- ------------ ------------------
 1    1    1.6.17           JI     FDO283707WH      Pass
 1    2    1.6.17           JI     FDO283707WH      Pass
 1    3    1.6.17           JI     FDO283707WG      Pass
 1    4    1.6.17           JI     FDO283707WG      Pass
```

Verify the software (Sw) and hardware (Hw) versions listed for each DPU.

Step 5 Verify the active DPU firmware package.

Example:

```
switch# show install active | grep dpu_fw
 dpu_fw-1.6.17-10.5.3.x86_64
```

Step 6 Verify the DPU initialization state in the DME.

Example:

```
switch# sh system internal dme running-config all dn sys/sas/dpu/ext | grep -i initState
"initState": "inventory-done",
```

Ensure the **initState** is **inventory-done**.

Step 7 Verify the DPU to service-ethernet interface mapping and DPU IP addresses.

Example:

```
switch# show dpu interface
            *************** DPU interface info **************

            Number of DPUs: 4
            Total number of DPU <-> NPU links: 4

              DPU-1:
              Number of links to NPU: 1
              NPU port number: 96
              Service Ethernet Interface(NPU): SEth1/1
              DPU-1 IP address: 169.254.28.1

              DPU-2:
              Number of links to NPU: 1
              NPU port number: 100
              Service Ethernet Interface(NPU): SEth1/2
              DPU-2 IP address: 169.254.24.1

              DPU-3:
              Number of links to NPU: 1
              NPU port number: 104
              Service Ethernet Interface(NPU): SEth1/3
              DPU-3 IP address: 169.254.36.1
```

```
DPU-4:
Number of links to NPU: 1
NPU port number: 108
Service Ethernet Interface(NPU): SEth1/4
DPU-4 IP address: 169.254.32.1
```

Verify the **Number of DPUs** and **Total number of DPU <-> NPU links**. For each DPU listed (e.g., **DPU-1**), ensure the **Number of links to NPU** is 1, the **Service Ethernet Interface(NPU)** maps to the correct interface (e.g., **SEth1/1** for DPU-1), and a unique **DPU-N IP address** is assigned (in the 169.254.x.x range).

After performing these checks, you should be able to confirm that the DPUs have powered up successfully and are in an operational state.

HSC to HSA Connection Issues

Diagnose and resolve issues preventing the Hypershield Agent on the switch from establishing a connection with the Hypershield Controller. The primary tool for troubleshooting HSC to HSA connectivity is the **service system hypershield test controller connection** command, which performs a series of checks.

Procedure

Step 1 Run the controller connection test utility.

Example:

```
switch# service system hypershield test controller connection
==============================
 Running config checks
 ==============================
 App Version: 1.1.0.rc.111          -> HSA running version
 Health Status: ok                  ->  HSA is healthy
 Using source-interface loopback100 for controller connection    -> Using IP from Lo100
 Agent registered with controller         -> HSA registered with HSC
 Controller connection token is configured      ->  OTP is configured
 Controller connection status: [success]    -> HSA connected with HSC
 ================================================
 Starting network connectivity checks on the switch    -> linux tests to verify network connectivity

 ================================================
 Using proxy http://proxy.esl.cisco.com:80 for controller connection
 Using curl to check connection to controller.prod.hypershield.engineering port 8880
 curl: (1) Received HTTP/0.9 when not allowed
 Curl successfully connected to controller.prod.hypershield.engineering:8880. Collect 'show tech
service-acceleration' to debug any agent connectivity failure. -> confirms the networking infra is
setup correct
```

This command performs several checks and indicates the status and potential reasons for connection failures.

In a successful connection, verify the following:

- The **Agent Status** indicates readiness, such as **firewall-ready, redirect-installed** (after full onboarding) or similar healthy states.
- The **Controller Connection Status** is **[success]**.

- Network connectivity checks (DNS resolution, ping, curl) are successful.

Example:

```
switch# service system Hypershield test controller connection
 ==============================
 Running config checks
 ==============================
 HypershieldAgent not running
 Configure 'service system hypershield'
```

If the agent is not running, the output explicitly states this.

To resolve this, ensure the **service system hypershield** command is configured.

Also, ensure the **source-interface loopback <idx>** is configured under **service system hypershield** and that the configured loopback interface has a valid /32 IP address in the default VRF.

Verify the agent state using **show system internal service-acceleration agent status**. It should show **started**.

Step 2 Interpret the output if the Firewall subsystem is not **in-service**.

Example:

```
switch(config-svc-sys)# service system hypershield test controller connection
  ==============================
  Running config checks
  ==============================
  Enable firewall service by configuring 'in-service'
```

If the firewall service is not enabled, the output prompts you to configure **in-service**.

To resolve this, configure **in-service** under the **service system hypershield service firewall** configuration mode.

Step 3 Interpret the output if the One-Time Password (OTP) is not configured.

Example:

```
switch(config-svc-sys-fw)# service system hypershield test controller connection
 ==============================
 Running config checks
 ==============================
 App Version: 1.1.0.rc.108
 Controller connection token missing. Configure using 'service system hypershield register <token>'
```

If the OTP has not been configured, the output indicates the missing token.

To resolve this, retrieve the OTP from the Hypershield Management Portal (HS MP) and configure it on the switch using the **service system hypershield register <token>** EXEC command.

You can verify if a token is configured (though not the token itself) using **sh system internal dme running-config all dn sys/sas/volatiledata/agent-hypershield | grep connToken**.

Step 4 Interpret the output if the HSA has not completed handshake and sync with all DPUs.

Example:

```
switch(config-svc-sys-fw)# service system hypershield test controller connection
==============================
Running config checks
==============================
App Version: 1.1.0.rc.108
Health Status: failed
Hypershield Agent <-> DPU handshake pending. Collect 'show tech-support service-acceleration'
==================================
HSA DPU connection diagnostics
```

```
================================
Client got: ok
Data:
IP 169.254.24.1, Number 2, InSync true, Healthy true
IP 169.254.28.1, Number 1, InSync false, Healthy true  -> DPU 1 InSync is false leading to failure

IP 169.254.32.1, Number 4, InSync true, Healthy true
IP 169.254.36.1, Number 3, InSync true, Healthy true
```

If the HSA cannot communicate or synchronize with one or more DPUs, the **Health Status** may show as **failed**, and the output will indicate a pending handshake or provide DPU diagnostics.

In the DPU diagnostics:

- **InSync false** indicates that the firewall policies are out of sync between the DPU and the HSA.
- **Healthy false** indicates that the periodic handshake between the DPU and the HSA is failing.

Collecting **show tech-support service-acceleration** is recommended for further debugging in this scenario.

Step 5 Interpret the output if required proxy configuration is missing or network connectivity issues exist.

Example:

```
switch(config-svc-sys-fw)# service system hypershield test controller connection
            ==============================
            Running config checks
            ==============================
            App Version: 1.1.0.rc.108
            Health Status: ok
            Using source-interface loopback1 for controller connection
            Controller connection token is configured
            Controller connection status: [init] -> HSA failed to connect to HSC
            Controller reason: [rpc error: code = Unavailable desc = connection error: desc =
"transport: Error while dialing: dial tcp 3.15.252.151:8880: i/o timeout"] -> Reason for the failureas
 reported by HSA
            ================================================
            Starting network connectivity checks on the switch
            ================================================
           Not using proxy for controller connection -> Indicates proxy not configured/used for the
 connection between HSA and HSC
            Using curl to check connection to controller.prod.hypershield.engineering port 8880
            curl: (28) Failed to connect to controller.prod.hypershield.engineering port 8880 after
 8753 ms: Connection timed out
            Connection failed. Curl error: 28. Running DNS and ping test-> unable to connect to HSC
 using curl indicating a network  issue either because of missing configuration or incorrecttopology

            nslookup www.google.com to check DNS resolution.
            Server:    171.70.168.183
            Address 1: 171.70.168.183 dns-sj.cisco.com
            Name:      www.google.com
            Address 1: 142.250.189.228 nuq04s39-in-f4.1e100.net
            Address 2: 2607:f8b0:4005:80e::2004 nuq04s39-in-x04.1e100.net
            nslookup www.google.com using DNS 171.70.168.183 success. -> DNS validation successful
indicating valid DNS and reachability to DNS
            Pinging controller.prod.hypershield.engineering to check network connectivity.
            PING controller.prod.hypershield.engineering (3.13.166.184) 56(84) bytes of data.
            --- controller.prod.hypershield.engineering ping statistics ---
            4 packets transmitted, 0 received, 100% packet loss, time 3033ms
            Could not ping controller.prod.hypershield.engineering. Checking for a valid route for
controller.prod.hypershield.engineering ->  reachability to HSC using ping failed
            3.12.93.183 via 36.36.36.1 dev Eth1-5 src 36.36.36.2 uid 0
            cache
            controller.prod.hypershield.engineering has a valid route on the host. -> indicates a
```

```
valid route for HSC  on the switch
            Validating container networking on the switch.
            Container networking on switch successful. -> confirms successful container networking
bring up
            controller.prod.hypershield.engineering has a valid route. Confirm if -> Failure could
potentially be because of any of the 3 reasons
            1) controller.prod.hypershield.engineering is valid and can be reached without a proxy
            2) source-interface 10.29.251.4 is routable in the network
            3) ICMP is not blocked in network.
```

If the HSA cannot reach the controller due to network issues or missing proxy configuration, the **Controller connection status** might be **[init]** or show an RPC error like "i/o timeout". The network connectivity checks using **curl** will fail.

If a proxy is required, ensure it is configured using **https-proxy <hostname|IP> port <port_num>** under the **service system hypershield** configuration mode.

The test utility also performs DNS and ping tests. Analyze their output to verify DNS resolution and basic IP reachability to the controller hostname/IP.

Step 6 (Optional) Test reachability from the HSA container using ping.

Example:

```
switch# service system hypershield test ping 171.70.33.31
PING 171.70.33.31 (171.70.33.31) 56(84) bytes of data.
64 bytes from 171.70.33.31: icmp_seq=1 ttl=53 time=2.93 ms
...
--- 171.70.33.31 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 1.540/2.357/2.931/0.525 ms
```

This command verifies basic IP connectivity from the HSA's perspective, as the ping is run from within the container. Specify the controller's IP address or hostname to ping.

By analyzing the output of the **service system hypershield test controller connection** command and performing additional ping tests from the HSA container, you can identify common configuration or network issues preventing the HSA from connecting to the HSC.

Network Configuration Before Firewall Bringup

Verify that the system is in the expected state when the firewall service is configured but not active. In this state, VRFs configured for service acceleration should be isolated, and traffic should not be redirected to the DPUs. If traffic is unexpectedly forwarded to the DPU or out of the box when VRFs are isolated, perform the checks in this section.

Procedure

Step 1 Verify that the **service firewall** is created and is in the **out-of-service** state.

Example:

```
switch# show run service-acceleration | grep firewall
            service firewall
```

This confirms the firewall service is configured.

Step 2 Check the operational status of the firewall service.

Example:

```
switch# show service-acceleration status
            Service System: hypershield
            Source Interface: loopback2 (10.29.251.7)
            Agent Status: firewall-agent-connection-pending,controller-connection-pending
            Agent Health Status: ok
            Controller Connection Status: init
            Services:
            Firewall: out-of-service --> by default out-of-service
```

The **Firewall** state should be **out-of-service**.

Step 3 Identify the VRFs enabled for service-acceleration.

Example:

```
switch# sh run service-acceleration
            !Command: show running-config service-acceleration
            !Running configuration last done at: Fri Apr 18 23:17:11 2025
            !Time: Fri Apr 18 23:17:14 2025
            version 10.5(3) Bios:version 01.09
            feature service-acceleration
            service system hypershield
            https-proxy proxy.esl.cisco.com port 80
            source-interface loopback2
            service firewall
            vrf yellow module-affinity 3
            vrf red module-affinity dynamic
```

Identify the VRFs configured for service acceleration by checking the **service firewall** section for lines starting with **vrf <name>**.

Step 4 Verify that the identified VRFs are in the **isolated** state.

Example:

```
switch# show service-acceleration status details
            Service System: hypershield
            Source Interface: loopback2 (10.29.251.7)
            Agent Status: firewall-agent-connection-pending,controller-connection-pending
            Agent Health Status: ok
            Controller Connection Status: init
            Services:
            Firewall: out-of-service
            VRF                              Operational State       Affinity(DPU)
            ================================================================
            red                              isolated                n/a  --> isolated state
            yellow                           isolated                n/a
```

Ensure the **Operational State** for each configured VRF is **isolated**.

Verify the routes advertised by the routing protocols for the VRF aligns with behavior supported during protocol isolation.

Note
VRF isolation behavior is currently supported in OSPF and BGP.

After performing these checks, you should be able to confirm that the VRFs configured for service acceleration are correctly identified as isolated by the system and routing protocols, which is the expected state before the firewall service is brought up (in-service).

NPU Packet Flows Before Firewall Bringup

When the firewall service is not active (out-of-service state), traffic intended for service-accelerated VRFs should be dropped or bypassed (for specific protocols like BFD or multicast), and not redirected to the DPUs. These steps verify this behavior. If traffic is unexpectedly forwarded to the DPU or out of the box despite correct VRF states and clean consistency checks, perform the verification steps in this section.

Procedure

Step 1 Run the consistency checker for a specific VRF to identify potential issues with traffic handling.

Example:

```
switch# show consistency-checker service-acceleration vrf red
            VRF 'red' is configured in Service-acceleration
            Executing Service-acceleration consistency check for VRF 'red'
            ****************************************
            SERVICE-ACCELERATION CONFIG CONSISTENCY
            ****************************************
            SUCCESS: VRF 'red' is configured with dynamic module affinity (0)
            SUCCESS: Module affinity matches for VRF 'red' - Configured: 0, Programmed: n/a
            ****************************************
            SAS OPERATIONAL CONSISTENCY
            ****************************************
            SUCCESS: VRF 'red' operational state 'isolated' is consistent with admin state
'out-of-service'
            ****************************************
            SAS REDIRECT CONSISTENCY
            ****************************************
            ****************************************
            POLICY ENFORCEMENT CONSISTENCY
            ****************************************
            SUCCESS: All data-plane interfaces for vrf red are present in configured interfaces of
Pass-1 ACL '__epbr_ip_dpu_inside'
          SUCCESS: All data-plane interfaces for vrf red are present in active interfaces for Pass-1
 ACL '__epbr_ip_dpu_inside'
            SUCCESS: All data-plane interfaces for vrf red are present in configured interfaces of
Pass-1 ACL '__epbr_ipv6_dpu_inside'
          SUCCESS: All data-plane interfaces for vrf red are present in active interfaces for Pass-1
 ACL '__epbr_ipv6_dpu_inside'
```

Replace **red** with the VRF name. This check verifies if the configuration and operational states are consistent and if the necessary policy enforcement points are in place. In the **out-of-service** state, verify the operational state is **isolated**.

Step 2 Verify that the internal policies generated for the intefaces of the VRF permit the BFD echo and multicast traffic and drop all other IP traffic.

Example:

```
switch# show access-lists __epbr_ip_dpu_inside  dynamic
            IP access list __epbr_ip_dpu_inside
            statistics per-entry
            1 permit udp any any eq 3785 ttl 255 [match=402393] --> Bypass for BFD
            2 permit ip any 224.0.0.0 15.255.255.255 [match=0]  --> Bypass for mcast
            4294967295 permit ip any any redirect Null0 [match=417359] --> catchall drop
```

This example shows the internally generated IPv4 policy. Verify the ACL counters to detect if any packets are matching the access-list.

Step 3 (Optional) Check counters on service-ethernet interfaces to verify no traffic is being sent to the DPU.

Example:

```
switch# Show interface service-ethernet 1/1 counters
            switch# Show interface service-ethernet 1/2 counters
            switch# Show interface service-ethernet 1/3 counters
            switch# Show interface service-ethernet 1/4 counters
```

Verify that the counters on these interfaces remain at zero or show only minimal control traffic, as traffic should not be sent to the **DPU** in the **out-of-service** state.

By performing these checks, you can confirm that the NPU is correctly handling traffic for service-accelerated VRFs in the out-of-service state by dropping or bypassing it according to the programmed Pass-1 ACLs, and that traffic is not being sent to the DPUs.

Firewall Bringup

Verify that the firewall service and configured VRFs have transitioned to the expected operational states after being enabled. The service firewall becomes ready asynchronously after the **in-service** command is executed. These steps confirm that the system is ready to attract and process traffic for firewalling.

Procedure

Step 1 Verify that the **in-service** command is configured for the firewall service.

Example:

```
switch# sh run service-acceleration | section 'service firewall'
service firewall
vrf red module-affinity dynamic
vrf yellow module-affinity 3
in-service
```

Verify that the **in-service** line is present under the **service firewall** configuration.

Step 2 Verify the final state of the HSA and the configured VRFs.

Example:

```
switch# sh service-acceleration status details
Service System: hypershield
Source Interface: loopback2 (10.29.251.7)
Agent Status: firewall-ready,redirect-installed    --> agent state after onboarding. There should
not be agent-connection-pending states seen here
Agent Health Status: ok
Controller Connection Status: success    --> controller connection successful
Services:
Firewall: in-service
VRF                             Operational State       Affinity(DPU)
============================================================
red                             forwarding ready        1  --> VRF fwd ready
yellow                          forwarding ready        3 --> DPU pinning must match static
configuration
```

This command provides a detailed status. Verify the following:

- The **Agent Status** indicates readiness (e.g., **firewall-ready,redirect-installed**).
- The **Controller Connection Status** is **success**.
- The **Firewall** service status is **in-service**.
- Each configured VRF shows an **Operational State** of **forwarding ready**.
- The **Affinity(DPU)** for each VRF shows a valid DPU number matching the configured affinity.

After performing these checks, you should be able to confirm that the firewall service is active and the system, including the HSA and configured VRFs, has reached the necessary operational state to begin processing traffic for firewalling.

Packet Forwarding

Verify the forwarding path for a specific IP flow, including service acceleration redirection, using the **show troubleshoot l3** command. This command checks routing information in various software and hardware tables and includes service acceleration consistency checks to help troubleshoot redirection issues.

Procedure

Step 1 Run the L3 troubleshooting command for a specific IPv4 flow.

Example:

```
switch# show troubleshoot l3 ipv4 181.1.1.3 src-ip 180.1.1.3 vrf red
            VRF 'red' is configured in Service-acceleration
            Executing Service-acceleration consistency check for VRF 'red'
            ****************************************
            SERVICE-ACCELERATION CONFIG CONSISTENCY
            ****************************************
            SUCCESS: Firewall admin state is in-service for vrf red
            SUCCESS: VRF 'red' is configured with dynamic module affinity (0)
            SUCCESS: Module affinity matches for VRF 'red' - Configured: 0, Programmed: 1
            ****************************************
            SAS OPERATIONAL CONSISTENCY
            ****************************************
            SUCCESS: VRF 'red' operational state 'forwarding ready' is consistent with admin state
'in-service'
            ****************************************
            SAS REDIRECT CONSISTENCY
            ****************************************
            SUCCESS: Pass-1 ACLs in VRF red have consistent redirect ports: SEth1/1.11
            SUCCESS: Pass-2 ACLs in VRF red have consistent redirect ports: SEth1/1.11
            ****************************************
            POLICY ENFORCEMENT CONSISTENCY
            ****************************************
            SUCCESS: All data-plane interfaces for vrf red are present in configured interfaces of
Pass-1 ACL '__epbr_ip_dpu_inside'
          SUCCESS: All data-plane interfaces for vrf red are present in active interfaces for Pass-1
 ACL '__epbr_ip_dpu_inside'
            SUCCESS: All data-plane interfaces for vrf red are present in configured interfaces of
Pass-1 ACL '__epbr_ipv6_dpu_inside'
```

```
          SUCCESS: All data-plane interfaces for vrf red are present in active interfaces for Pass-1
ACL '__epbr_ipv6_dpu_inside'
           ************************************
           CHECKING HARDWARE ASIC TYPE
           ************************************
           SWITCH TYPE: TOR

           1. CHECK ROUTE IN PI RIB
           ************************************
           <<snip>>
           ************************************
           2.  CHECK ROUTE IN PD FIB
           ************************************
           <<snip>>
           ************************************
           CHECK ROUTE IN UFIB
           ************************************
           <<snip>>
           ************************************
           3. CHECK HOST ROUTE IN FIB AND HARDWARE
           ************************************
           <<snip>>
           ************************************
           4. CHECK ROUTE,NEXTHOP IN HARDWARE
           ************************************
           <<snip>>
           ************************************
           RUNNING CONSISTENCY CHECKER
           ************************************
           ************************************
           CHECKING HARDWARE ASIC TYPE
           ************************************
           Please wait, consistency checker may take a while...
           Consistency checker passed for 181.1.1.0/24
```

Specify the destination IP (**181.1.1.3**), source IP (**180.1.1.3**), and VRF name (**red**) for the flow you are troubleshooting.

Analyze the output, which includes:

- Service acceleration consistency checks for the specified VRF.
- Route lookups in the Protocol Independent RIB (PI RIB), Protocol Dependent FIB (PD FIB), and Unified FIB (UFIB).
- Checks for host routes and next-hops in hardware.
- A final consistency checker run.

Verify that consistency checks show **SUCCESS** messages and that route lookups identify the expected next-hop or redirection path.

Step 2 (Optional) Run the L3 troubleshooting command for a specific IPv6 flow.

Example:

```
switch# show troubleshoot l3 ipv6 1:11:: src-ip 1:10:: vrf red
```

Use this command to troubleshoot IPv6 packet forwarding issues, replacing the IP addresses and VRF name as needed.

If the **show troubleshoot l3** command completes successfully and reports "Consistency checker passed" for the route, it indicates that the system has the necessary routing and service acceleration configuration in place to forward the specified packet flow, including the redirection to the DPU.

HSA to NXOS Interactions

Verify that the Hypershield Agent (HSA) has successfully pushed the necessary configuration objects to NX-OS via the Data Management Engine (DME) for traffic redirection using EPBR. If traffic is arriving at the switch and dropping or bypassing DPUs despite clean consistency checks, verify the redirection information in the switch using the steps in this section.

Procedure

Verify if the agent has sent the redirect policies and services configuration to NX-OS.

Example:

```
switch# show system internal service-acceleration dynamic configuration
!Command: show system internal service-acceleration dynamic configuration
!Running configuration last done at: Sat Apr 19 01:01:59 2025
!Time: Sat Apr 19 01:20:32 2025
version 10.5(3) Bios:version 01.09
vrf context red
epbr policy __red_dpu_redir   --> enforce redirection policy on VRF
vrf context yellow
epbr policy __yellow_dpu_redir
epbr service __red_dpu_redir type dpu
vrf red
service-end-point module 1 vlan 11   --> VRF red traffic to redirect to DPU 1 with encap VLAN 11
epbr service __yellow_dpu_redir type dpu
vrf yellow
service-end-point module 3 vlan 10
epbr policy __red_dpu_redir
statistics
match ipv6 address __dpu_ipv6_redir    --> match on all ipv6 traffic
10 set service __red_dpu_redir fail-action drop      --> redirect to service
match ip address __dpu_redir     --> match on all ipv4 traffic
10 set service __red_dpu_redir fail-action drop      --> redirect to service
epbr policy __yellow_dpu_redir
statistics
match ipv6 address __dpu_ipv6_redir
10 set service __yellow_dpu_redir fail-action drop
match ip address __dpu_redir
10 set service __yellow_dpu_redir fail-action drop
```

Verify the dynamic EPBR configuration pushed by the Hypershield Agent. Ensure that EPBR policies are associated with the correct VRFs and that EPBR services define the correct DPU module and encapsulation VLAN ID for each VRF.

By performing these checks, you can confirm that the Hypershield Agent has successfully pushed the necessary EPBR configuration objects to NX-OS via DME, defining how traffic for service-accelerated VRFs should be redirected to the appropriate DPUs.

NPU to DPU Redirection Issues

Verify that EPBR redirection policies are correctly programmed in the NPU hardware (TCAM) and that traffic is successfully redirected to the appropriate DPU via service-ethernet subinterfaces. These checks help identify the cause if traffic is arriving but not being redirected to the DPU or is sent to the wrong DPU, even if policies are correctly pushed by the agent.

Procedure

Step 1 Identify the service sub-interfaces created for each VRF to redirect the VRF traffic to the DPU and verify their status.

Example:

```
switch# show service-acceleration redirect-policy brief
VRF                              AF Type Interface[Status]  Affinity Redirect Status
================================================================
red                              IPv4    SEth1/1.11[UP]     1        Enabled
yellow                           IPv4    SEth1/3.10[UP]     3        Enabled
red                              IPv6    SEth1/1.11[UP]     1        Enabled
yellow                           IPv6    SEth1/3.10[UP]     3        Enabled
```

Identify the service sub-interface (**Interface[Status]**) created for each VRF and address family. Verify the interface status is **UP** and the **Redirect Status** is **Enabled**.

Step 2 Verify the configuration details of a specific service-ethernet sub-interface.

Example:

```
switch# sh interface service-ethernet 1/1.11 br
 ------------------------------------------------------------------------------
 Service         VLAN    Type Mode   Status  Reason                Speed     Port
 Ethernet                                                                    Ch #
 ------------------------------------------------------------------------------
 SEth1/1.11      11      eth  routed up      none                   200G(D) --
```

Specify the sub-interface (**1/1.11**) identified in the previous step. Verify that the **VLAN** ID matches the one configured in the EPBR service for the VRF, the **Mode** is **routed**, and the **Status** is **up**.

Step 3 Verify the detailed policy programming information in EPBR for a specific VRF.

Example:

```
switch# show service-acceleration redirect-policy vrf red
 VRF: red
 Policy-map: __red_dpu_redir
 Default Traffic Action: Redirect
 Match clause:
 ip address(access-lists): __dpu_ipv6_redir
 action: Redirect
 status(inside): CREATED/ENABLED         status(post_fwd): CREATED/ENABLED
 service __red_dpu_redir, sequence 10, fail-action Drop
 service-module 1 vlan 11 [UP] [SEth1/1.11]
 Match clause:
 ip address(access-lists): __dpu_redir
 action: Redirect
 status(inside): CREATED/ENABLED         status(post_fwd): CREATED/ENABLED
 service __red_dpu_redir, sequence 10, fail-action Drop
 service-module 1 vlan 11 [UP] [SEth1/1.11]
```

Replace **red** with the VRF name. Verify that the **status(inside)** and **status(post_fwd)** for the match clauses show **CREATED/ENABLED**. Confirm that the **service-module**, **vlan**, and service interface (**SEth1/1.11**) information is correct and matches the expected DPU and sub-interface for this VRF.

Step 4 Verify the packet counters matching the internally generated policies for the VRF to redirect to the DPU.

Example:

(IPv4 traffic)

```
switch# show access-lists  __epbr_ip_dpu_inside dynamic
IP access list __epbr_ip_dpu_inside  --> IPv4 traffic
statistics per-entry
1 permit udp any any eq 3785 ttl 255 [match=129782]
2 permit ip any 224.0.0.0 15.255.255.255 [match=0]
1701 permit ip any any nve vni 104 redirect Service-Ethernet1/3.10 [match=43683]
6601 permit ip any any nve vni 6 redirect Service-Ethernet1/1.11 [match=43618]  --> VNI matches IPv4
 table ID for VRF, Redirect interface matches
4294967295 permit ip any any redirect Null0 [match=47199]
```

Example:

(IPv6 traffic)

```
switch# show access-lists  __epbr_ipv6_dpu_inside dynamic
IPv6 access list __epbr_ipv6_dpu_inside --> IPv6 traffic
statistics per-entry
1 permit udp any any eq 3785 ttl 255 [match=59751]
2 permit ipv6 any ff00:: ff:ffff:ffff:ffff:ffff:ffff:ffff:ffff [match=0]
1401 permit ipv6 any any nve vni 104 redirect Service-Ethernet1/3.10 [match=47256]
6301 permit ipv6 any any nve vni 6 redirect Service-Ethernet1/1.11 [match=46572]  --> VNI matches
IPv4 table ID for VRF, Redirect interface matches
4294967295 permit ipv6 any any redirect Null0 [match=30781]
```

Identify the rules relevant to the VRF of interest by identifying the table id for the VRF via **show vrf <> detail** which is used as the VNI in the above rules. Verify that the rules show redirection to the right subinterface for the VRF. Verify the packet hit counters for the VRF rules.

Step 5 (Optional) Verify resource utilization for Service-acceleration redirect policies.

Example:

```
switch# show system internal access-list resource utilization | grep -i pass
  Ingress SVC Redir L3 Policy Pass-1-IPV4                    5       7115    0.07
  Ingress SVC Redir L3 Policy Pass-1-IPV6                    5       3555    0.14
  Ingress SVC Redir L3 Policy Pass-2-IPV4                    6       7114    0.08
  Ingress SVC Redir L3 Policy Pass-2-IPV6                    6       3554    0.16
  LBL AO    Ingress IPv4 SVC Redir L3 Policy Pass-1          2       125     1.57
  LBL AP    Ingress IPv6 SVC Redir L3 Policy Pass-1          2       125     1.57
  LBL AQ    Ingress IPv4 SVC Redir L3 Policy Pass-2          2       125     1.57
  LBL AR    Ingress IPv6 SVC Redir L3 Policy Pass-2          2       125     1.57
```

Verify hardware resource usage for ACL policies, including **Pass-1** and **Pass-2** redirect policies. High utilization or errors may indicate resource exhaustion.

By performing these checks, you can verify that the NPU is correctly programmed to redirect traffic for service-accelerated VRFs to the appropriate DPU via the service-ethernet sub-interfaces, and confirm that traffic is hitting the expected hardware rules.

Post DPU Packet Flows

Verify that traffic returning from the DPU is correctly processed by the NPU for forwarding towards its destination by checking the Pass-2 ACLs applied to service-ethernet interfaces and verifying interface configuration details. If traffic is dropping in the NPU or not routing after returning from the DPU, perform the verification steps in this section.

Procedure

Step 1 Verify the packet counters matching the internally generated policies for the VRF to redirect to the DPU for the second pass of inspection, before the flows are learnt.

Example:

(IPv4 traffic)

```
switch# sh access-lists __epbr_ip_dpu_post_fwd dynamic
  IP access list __epbr_ip_dpu_post_fwd --> IPv4 traffic
  statistics per-entry
  1701 permit ip any any nve vni 104 redirect Service-Ethernet1/3.10 [match=0]
  6601 permit ip any any nve vni 6 redirect Service-Ethernet1/1.11 [match=0]
  4294967295 permit ip any any [match=0]
```

Example:

(IPv6 traffic)

```
switch# sh access-lists __epbr_ipv6_dpu_post_fwd dynamic
 IPv6 access list __epbr_ipv6_dpu_post_fwd --> IPv6 traffic
 statistics per-entry
 1401 permit ipv6 any any nve vni 104 redirect Service-Ethernet1/3.10 [match=0]
 6301 permit ipv6 any any nve vni 6 redirect Service-Ethernet1/1.11 [match=0]  --> VNI matches VRF
table ID and redirect interfaces matches for VRF
 4294967295 permit ipv6 any any [match=0]
```

Identify the rules relevant to the VRF of interest by identifying the table ID for the VRF via **show vrf <> detail**, which is used as the VNI in the above rules. Verify that the rules include permit entries matching traffic returning from the DPU and redirect it to the correct service sub-interface for the VRF. Confirm the packet hit counters for these VRF rules.

Step 2 Show statistics for DPU connected service-ethernet ports.

Example:

```
switch# Show interface service-ethernet 1/1-4
            switch# Show interface service-ethernet 1/2 counters
            switch# show interface service-port-channel 1-4
            switch# show int service-ethernet 1/2 counters detailed
            switch# Show interface service-ethernet <subinterface> counters
            switch# show int service-ethernet <> counters errors
            switch# show int service-ethernet 1/1-4 counters brief
```

These commands provide various statistics for the service-ethernet interfaces connected to the DPUs, including packet/byte counters, detailed stats, error stats, and average rates. Checking these counters helps determine if traffic is reaching or leaving the DPUs as expected.

By performing these checks, you can verify that the NPU has correctly programmed the Pass-2 ACLs to process traffic returning from the DPU and that the service-ethernet interfaces and subinterfaces are configured with the correct parameters for forwarding.

Packet Tracer Support

Use the Packet Tracer utility to capture and analyze packets at different points in the end-to-end NPU to DPU to NPU packet flow. This allows for detailed troubleshooting of traffic handling at various stages.

The following steps outline how to capture packets at specific points:

- NPU ingress
- NPU to DPU TX
- DPU to NPU RX - first packet
- DPU to NPU RX - subsequent packets
- NPU egress

Procedure

Step 1 Capture packets received on the ingressing L3 interface (NPU ingress).

Example:

```
switch# packet-trace
  switch(packet-trace)# trigger init rxpp
  switch(packet-trace-init)# packet-format eth-ipv4-tcp
  switch(packet-trace-init-pkt)# set outer ipv4 src-ip 10.1.0.2 dst-ip 16.1.0.2
  switch(packet-trace-init-pkt)# start
  ...
  switch(packet-trace-init-pkt)# status
  switch(packet-trace-init-pkt)# report
  switch(packet-trace-init-pkt)# exit
```

This captures packets received by the NPU on the ingressing Layer 3 interface before redirection to the DPU.

Step 2 Capture packets transmitted from the NPU to the DPU on the service-ethernet interface (SETH TX).

Example:

```
switch# packet-trace
  switch(packet-trace)# trigger init txpp
  switch(packet-trace-init)# packet-format eth-ipv4-tcp
  switch(packet-trace-init-pkt)# set outer ipv4 src-ip 10.1.0.2 dst-ip 16.1.0.2
  switch(packet-trace-init-pkt)# start
  ...
  switch(packet-trace-init-pkt)# status
  switch(packet-trace-init-pkt)# report
  switch(packet-trace-init-pkt)# exit
```

This captures packets transmitted by the NPU towards the DPU on the service-ethernet interface.

Note
The snapshot is taken before the DPU header encapsulation happens, so DPU header details cannot be traced or captured at this point.

Step 3 Capture the first packet received by the NPU from the DPU with the pass bit set (SETH RX dpu pass 0).

Example:

```
switch# packet-trace
  switch(packet-trace)# trigger init rxpp
  switch(packet-trace-init)# packet-format eth-dpu-ipv4-tcp
  switch(packet-trace-init-pkt)# set dpu pass-bit 0 service-vlan 10
  switch(packet-trace-init-pkt)# start
  ...
  switch(packet-trace-init-pkt)# status
  switch(packet-trace-init-pkt)# report
  switch(packet-trace-init-pkt)# exit
```

This captures the initial packets received by the NPU from the DPU on the service-ethernet interface, specifically those marked with DPU pass bit 0 (typically the first packet of a flow punted to the DPU's control plane).

Note
The snapshot is taken before DCPU decap happens, but you can get insights into the received DPU header.

Step 4 Capture subsequent packets received by the NPU from the DPU with the pass bit set (SETH RX dpu pass 1).

Example:

```
switch# packet-trace
  switch(packet-trace)# trigger init rxpp
  switch(packet-trace-init)# packet-format eth-dpu-ipv4-tcp
  switch(packet-trace-init-pkt)# set dpu pass-bit 1 service-vlan 10
  switch(packet-trace-init-pkt)# start
  ...
  switch(packet-trace-init-pkt)# status
  switch(packet-trace-init-pkt)# report
  switch(packet-trace-init-pkt)# exit
```

This captures subsequent packets for a flow received by the NPU from the DPU on the service-ethernet interface, typically those processed directly by the DPU's data plane (P4) and marked with DPU pass bit 1.

Note
The snapshot is taken before DCPU decap happens, but you can get insights into the received DPU header.

Step 5 (Optional) Capture specific packets received by the NPU from the DPU based on IP addresses (SETH RX dpu pass 1 - specific flow).

Example:

```
switch# packet-trace
  switch(packet-trace)# trigger init rxpp
  switch(packet-trace-init)# packet-format eth-dpu-ipv4-tcp
  switch(packet-trace-init-pkt)# set dpu pass-bit 1 service-vlan 10
  switch(packet-trace-init-pkt)# set outer ipv4 src-ip 10.1.0.2 dst-ip 16.1.0.2
  switch(packet-trace-init-pkt)# start
  ...
  switch(packet-trace-init-pkt)# status
  switch(packet-trace-init-pkt)# report
  switch(packet-trace-init-pkt)# exit
```

This captures subsequent packets for a specific flow received by the NPU from the DPU. It combines the DPU filters with outer IP address filters.

Step 6 Capture packets transmitted from the NPU to the egress L3 interface after routing (NPU egress).

Example:

```
switch# packet-trace
  switch(packet-trace)# trigger init txpp
  switch(packet-trace-init)# packet-format eth-dpu-ipv4-tcp
  switch(packet-trace-init-pkt)# set outer ipv4 src-ip 10.1.0.2 dst-ip 16.1.0.2
  switch(packet-trace-init-pkt)# start
  ...
  switch(packet-trace-init-pkt)# status
  switch(packet-trace-init-pkt)# report
  switch(packet-trace-init-pkt)# exit
```

This captures packets transmitted by the NPU towards the egress Layer 3 interface after being processed and routed post-DPU.

Note
Note that the snapshot is taken before the DPU header encapsulation happens, so DPU header details cannot be traced or captured at this point.

After performing these packet capture steps, you can analyze the captured packets to gain insight into how traffic is being processed at different points in the NPU to DPU to NPU flow.