Revised: September 9, 2026

Troubleshooting Smart Switches in DPU Security mode

Troubleshoot Smart Switches in DPU Security Mode

This article covers the end-to-end packet flow, spanning from the NPU hardware programming to DPU-based inspection and High Availability (HA) state synchronization. This article also provides systematic procedures for identifying, isolating, and resolving issues within the Smart Switch environment when integrated with Hypershield.

As Hypershield introduces advanced traffic redirection and inspection capabilities—leveraging Data Processing Units (DPUs) and redirection from NPU—troubleshooting requires a multi-layered approach.

The contents in this article are intended for users who have both network-admin and netsecops-admin roles. For more information about these user roles, refer to the User Accounts and RBAC for Smart Switches chapter in the Cisco Smart Switches Configuration Guide for DPU Security Mode. It is assumed that the users have a working knowledge of NX-OS, VRF-lite architectures, and the service-acceleration feature.

Note

The commands and outputs provided herein are based on the current NX-OS Release. Always ensure your environment is running the supported software version before performing deep-dive troubleshooting.

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.

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     1.150.2      JI         FDO283707WH   Pass
1   2     1.150.2      JI         FDO283707WH   Pass
1   3     1.150.2      JI         FDO283707WG   Pass
1   4     1.150.2      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-1.150.2-10.6.3q.x86_64
```

Step 6 Verify the DPU initialization state in the DME.

Example:

```
switch# show 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 (for example, DPU-1), ensure the Number of links to NPU is 1, the Service Ethernet Interface (NPU) maps to the correct interface (for example, SEth1/1 for DPU-1), and a unique DPU-N IP address is assigned (in the 169.254.x.x range).

What to do next

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

Hypershield Controller to Hypershield Agent Connection Issues

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

Step 1 Run the controller connection test utility.

Example:

```
switch# service system hypershield test controller connection
==============================
 Running config checks
==============================
Using source-interface loopback100 for controller connection
Controller connection token is configured
Controller connection status: [success]
Controller reason: [connected ok with Hypershield controller]
Controller endpoint: 172.31.147.51:6443
================================================
Starting network connectivity checks on the switch
================================================
Not using proxy for controller connection
Using curl to check connection to 172.31.147.51 port 6443
Curl successfully connected to 172.31.147.51:6443.
Collect 'show tech service-acceleration' to debug any agent connectivity failure
```

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 that **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 the **show system internal service-acceleration agent status** command. 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 token is not configured or an invalid token is configured.

Example:

```
switch(config-svc-sys-fw)# service system hypershield test controller connection

==============================
 Running config checks
==============================
Controller connection token Invalid! 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 **show system internal dme running-config all dn sys/sas/volatiledata/agent-hypershield | grep connToken**.

Step 4 This is the output if the HSA has completed handshake and sync with all DPUs.

Example:

```
switch# service system hypershield test controller connection

==============================
 Running config checks
==============================
Hypershield Agent <-> DPU handshake pending. Collect 'show tech-support service-acceleration'
```

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 the output of the **show tech-support service-acceleration** command is recommended for further debugging in this scenario.

Step 5 When there is no route to the controller the output shows the detailed network connectivity checks and routing verification.

Example:

```
switch(config-if)# service system hypershield test controller connection

==============================
 Running config checks
==============================
Using source-interface loopback100 for controller connection
Controller connection token is configured
Controller connection status: [success]
Controller reason: [connected ok with Hypershield controller]
Controller endpoint: 172.31.147.51:6443
```

```
================================================
 Starting network connectivity checks on the switch
================================================
Not using proxy for controller connection
Using curl to check connection to 172.31.147.51 port 6443
Failed to connect to host.

Running ping tests to check connectivity.
Pinging 172.31.147.51 to check network connectivity.
PING 172.31.147.51 (172.31.147.51) 56(84) bytes of data.
From 127.253.254.1 icmp_seq=1 Destination Net Unreachable

--- 172.31.147.51 ping statistics ---
4 packets transmitted, 0 received, +1 errors, 100% packet loss, time 3037ms

Could not ping 172.31.147.51.

Checking for a valid route for 172.31.147.51
RTNETLINK answers: Network is unreachable

Could not reach 172.31.147.51 from host.
Check 'show ip route 172.31.147.51' for a valid route

switch(config-if)# show ip route 172.31.147.51
IP Route Table for VRF "default"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
Route not found
```

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.

Load Balance mode verification

Verify the configured and operational load-balance configuration. Verify that the service port-channel interfaces for the load-balance mode have been created as expected.

These sections explain troubleshooting for:

- Symmetric hash
- Pinning load balance mode

Symmetric hash

Step 1 Identify the load-balance configuration for service-acceleration. In this example, the load-balance configuration is at its default value of symmetric-hash.

Example:

```
switch# show run service-acceleration
!Command: show running-config service-acceleration
!Running configuration last done at: Fri May  8 07:35:41 2026
!Time: Sat May  9 23:02:10 2026
version 10.6(3q) Bios:version 01.17
feature service-acceleration
service system hypershield
  source-interface loopback2
  service firewall
    vrf tenant1 module-affinity dynamic
    vrf tenant2 module-affinity dynamic
    vlan id 2003,2005 bridged-traffic module-affinity dynamic
```

Step 2 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 3 Verify that the service firewall is created. Verify that this shows correctly as out-of-service and the load-balance mode is set to the default mode of symmetric-hash.

Example:

```
show run service-acceleration  | grep firewall
  service firewall
switch# show service-acceleration status
  Service System: hypershield
    Source Interface: loopback2 (10.29.251.4)
    DPU Load-balance Mode: symmetric-hash  --> default load-balance mode
    Agent Status: firewall-disable
    Agent Health Status: ok
    Controller Connection Status: init
    Agent HA Status: not-initialized
    Services:
      Firewall: out-of-service             --> by default out-of-service
```

Step 4 Verify service port-channel creation and membership for symmetric-hash.

Example:

```
switch(config)# show port-channel summary interface service-port-channel 512
<<snip>>
--------------------------------------------------------------------------------
Group Port-Channel Type     Protocol  Member Ports
--------------------------------------------------------------------------------
512   SPo512(RU)  Eth       NONE      SEth1/1(P)   SEth1/2(P)   SEth1/3(P)
                                     SEth1/4(P)
```

Note In the above example, only `SPO 512` has all 4 Service-Ethernet links pertaining to each DPU.

Step 5 Use the consistency-checker for port-channel membership for any service port-channel issues.

Example:

```
switch# show consistency-checker membership port-channels interface service-port-channel 512
Checks: Pif, vifstate, localmultipath, ucpcconfig and ucpcmember tables.
Performing Consistency-checker for Service-Port-Channel512
Checking hardware for Module 1
Module 1 Consistency Check: PASSED
['Service-Ethernet1/1', 'Service-Ethernet1/2', 'Service-Ethernet1/3', 'Service-Ethernet1/4']
```

Step 6 Verify that the Hypershield Agent has identified the load-balance mode correctly.

Example:

```
switch# app-hosting connect appid HypershieldAgent session /usr/src/app/agwctl device show | grep
'LB Mode'
  LB Mode:               symmetric-hash
```

Pinning load balance mode

Step 1 Identify the load-balance configuration for service-acceleration. In this example, the load-balance configuration is configured as pinning.

Example:

```
switch(config)# show run service-acceleration
!Command: show running-config service-acceleration
!Running configuration last done at: Sun May 10 00:14:30 2026
!Time: Sun May 10 00:18:06 2026
version 10.6(3q) Bios:version 01.17
feature service-acceleration
service dpu load-balance pinning         --> Pinning load-balance mode
service system hypershield
  source-interface loopback2
  service firewall
    vrf tenant1 module-affinity 1
    vrf tenant2 module-affinity dynamic
    vlan id 2003 bridged-traffic module-affinity 3
    vlan id 2005 bridged-traffic module-affinity dynamic
```

Step 2 Verify that the service firewall is created. Verify that this shows correctly as out-of-service and that the load-balance mode shows the pinning mode.

Example:

```
show run service-acceleration  | grep firewall
  service firewall
switch# show service-acceleration status
  Service System: hypershield
Source Interface: loopback2 (10.29.251.4)
    DPU Load-balance Mode: pinning         -->  Pinning load-balance mode
    Agent Status: firewall-disable
    Agent Health Status: ok
    Controller Connection Status: init
    Agent HA Status: not-initialized
    Services:
      Firewall: out-of-service             --> by default out-of-service
```

Step 3 Verify service per-DPU port-channel creation and membership for pinning mode. The output for N9324C-SE1U that is captured here indicates that four service port-channels are created, one for each DPU. Each one of these port-channels has a single service-ethernet interface as its member link.

Example:

```
switch(config)# show port-channel summary interface service-port-channel 1
<<snip>>
--------------------------------------------------------------------------------
Group Port-Channel Type     Protocol  Member Ports
--------------------------------------------------------------------------------
SPo1(RU)    Eth      NONE      SEth1/1(P)
--> Service-ethernet interfaces pertaining to each DPU in per-DPU SPOs
switch(config)# show port-channel summary interface service-port-channel 2
<<snip>>
--------------------------------------------------------------------------------
Group Port-Channe Type     Protocol  Member Ports
--------------------------------------------------------------------------------
2     SPo2(RU)    Eth      NONE      SEth1/2(P)
switch(config)# show port-channel summary interface service-port-channel 3
<<snip>>
--------------------------------------------------------------------------------
Group Port-Channel Type     Protocol  Member Ports
--------------------------------------------------------------------------------
3     SPo3(RU)    Eth      NONE      SEth1/3(P)
switch(config)# show port-channel summary interface service-port-channel 4
<<snip>>
--------------------------------------------------------------------------------
Group Port-Channel Type     Protocol  Member Ports
--------------------------------------------------------------------------------
4     SPo4(RU)    Eth      NONE      SEth1/4(P)
```

On N9348Y2C6D-SE1U, two service-port-channels, one per DPU, are created. Each service port-channel has two service-ethernet interfaces as its member links.

Step 4 Use the consistency-checker for port-channel membership for any service port-channel issues.

Example:

```
switch# show consistency-checker membership port-channels interface service-port-channel 1
Checks: Pif, vifstate, localmultipath, ucpcconfig and ucpcmember tables.
Performing Consistency-checker for Service-Port-Channel1
Checking hardware for Module 1
Module 1 Consistency Check: PASSED
['Service-Ethernet1/1’]
```

Step 5 Verify that the Hypershield Agent has identified the load-balance mode correctly.

Example:

```
switch# app-hosting connect appid HypershieldAgent session /usr/src/app/agwctl device show | grep
'LB Mode'
  LB Mode:               pinning
```

What to do next

After performing these checks, you can confirm that the system has detected and enabled the correct load-balance mode as per the configuration.

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 the VLANs should be set to drop. Traffic should not be redirected to the DPUs and must be dropped. If traffic is unexpectedly forwarded to the DPU or out of the box when VRFs are isolated or when VLANs are set to drop, perform the checks in this section.

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)
    DPU Load-balance Mode: symmetric-hash
    Agent Status: firewall-disable
    Agent Health Status: ok
    Controller Connection Status: init
    Agent HA Status: not-initialized
    Services:
      Firewall: out-of-service    --> by default out-of-service
```

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

Step 3 Identify the VRFs and VLANs enabled for service-acceleration.

Example:

```
switch# show  running-config service-acceleration

!Command: show running-config service-acceleration
!Running configuration last done at: Thu May  7 06:04:18 2026
!Time: Thu May  7 06:04:33 2026

version 10.6(3q) Bios:version 01.17
feature service-acceleration

service system hypershield
  source-interface loopback2
  service firewall
    vrf tenant1 module-affinity dynamic
    vrf tenant2 module-affinity dynamic
    vlan id 2003,2005 bridged-traffic module-affinity dynamic
```

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

Identify the VLANs configured for service acceleration by checking the `service firewall` section for lines starting with `vlan <id>`.

Step 4 Verify that the identified VRFs are in `isolated` state and that the identified VLANs are set to drop.

Example:

```
switch# show service-acceleration status details
  Service System: hypershield
    Source Interface: loopback2 (10.29.251.4)
    DPU Load-balance Mode: symmetric-hash
    Agent Status: firewall-disable
    Agent Health Status: ok
    Controller Connection Status: init
    Agent HA Status: not-initialized
    Services:
      Firewall: out-of-service
```

```
VRF                  Operational State       Affinity(DPU)
============================================================
tenant1               isolated                n/a      --> isolated state
tenant2               isolated                n/a

VLAN                 Operational State       Affinity(DPU)
============================================================
2003                  VLAN set to drop        n/a
2005                  VLAN set to drop        n/a
```

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 for OSPF and BGP.

What to do next

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). You can verify the expected states for VLANs configured for service-acceleration.

NPU packet flows before firewall bringup

When the firewall service is not active (out-of-service state), traffic intended for firewall 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. Traffic intended for Firewall VLANs should be dropped or bypassed (for multicast and broadcast traffic) and not redirected to the DPUs.

Step 1 Run the Consistency checker for the VRF first for any issues where traffic is expected to land on the switch, but is not dropping or directly routing.

Example:

```
switch(config)# show consistency-checker service-acceleration vrf tenant1
=======================================
PROCESSING VRF CONSISTENCY CHECK REQUEST
=======================================
VRF 'tenant1' is configured in Service-acceleration
Executing Service-acceleration consistency check for VRF 'tenant1'
***************************************
SERVICE-ACCELERATION CONFIG CONSISTENCY
***************************************
 SUCCESS: Firewall admin state is out-of-service.
 SUCCESS: VRF 'tenant1' is configured with dynamic module affinity (0)
 SUCCESS: Module affinity matches for VRF 'tenant1' - Configured: 0, Programmed: n/a
***************************************
SAS OPERATIONAL CONSISTENCY
***************************************
 SUCCESS: VRF 'tenant1' operational state 'isolated' is consistent with admin state 'out-of-service'
***************************************
SAS REDIRECT CONSISTENCY
***************************************
 SUCCESS: Pass-1 ACLs in VRF tenant1 have consistent redirect ports: None
 SUCCESS: Pass-2 ACLs in VRF tenant1 have consistent redirect ports: None
 WARNING: ACL: No interfaces found with redirect rules for VRF tenant1
***************************************
POLICY ENFORCEMENT CONSISTENCY
***************************************
```

```
 SUCCESS: All data-plane interfaces for vrf tenant1 are present in configured interfaces of Pass-1
ACL '__epbr_ip_dpu_inside'

 SUCCESS: All data-plane interfaces for vrf tenant1 are present in active interfaces for Pass-1 ACL
 '__epbr_ip_dpu_inside'

 SUCCESS: All data-plane interfaces for vrf tenant1 are present in configured interfaces of Pass-1
ACL '__epbr_ipv6_dpu_inside'

 SUCCESS: All data-plane interfaces for vrf tenant1 are present in active interfaces for Pass-1 ACL
 '__epbr_ipv6_dpu_inside'
```

Replace `tenant1` 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 Run the Consistency checker for the VLAN first for any issues where traffic is expected to land on the switch, but is not dropping or directly forwarding

Example:

```
switch# show consistency-checker service-acceleration vlan 2005
========================================
PROCESSING VLAN CONSISTENCY CHECK REQUEST
========================================
VLAN '2005' is configured in Service-acceleration
Executing Service-acceleration consistency check for VLAN '2005'
****************************************
SERVICE-ACCELERATION CONFIG CONSISTENCY
****************************************
 SUCCESS: Firewall admin state is out-of-service.
****************************************
SAS MODULE AFFINITY CONSISTENCY
****************************************
 SUCCESS: VLAN '2005' is configured with dynamic module affinity (0)
 SUCCESS: Module affinity matches for VLAN '2005' - Configured: 0, Programmed: n/a
****************************************
SAS OPERATIONAL CONSISTENCY
****************************************
 SUCCESS: VLAN '2005' operational state 'VLAN set to drop' is consistent with admin state
'out-of-service'
****************************************
SAS VLAN ACCESS MAP CONSISTENCY
****************************************
 SUCCESS: VLAN access map '__epbr_ip_dpu_inside_drop' for Vacl '__epbr_dpu_inside_catchall' and action
 'drop' is consistent
****************************************
SAS VLAN FILTER CONSISTENCY
****************************************
 SUCCESS: VLAN filter __epbr_dpu_inside_catchall for VLAN '2005'
****************************************
SAS VLAN TCAM REDIRECT CONSISTENCY
****************************************
 WARNING: No redirect entries found for VLAN 2005
```

Replace `2005` with the VLAN number. This check verifies if the configuration and operational states are consistent and the requirement policy enforcement is in place. The operational state for VLANs is set to drop when the firewall is in the `out-of-service` state.

Step 3 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 4 Verify that the internal policies generated for the VLAN permit multicast and broadcast traffic and drop all IP traffic.

Example:

```
switch# show vlan access-map __epbr_dpu_inside_catchall
Vlan access-map __epbr_dpu_inside_catchall 1
        match ip: __epbr_ip_dpu_inside_bypass
        action: forward         --> Regular forwarding for IP broadcast and multicast traffic
        statistics per-entry
Vlan access-map __epbr_dpu_inside_catchall 2
        match ip: __epbr_ip_dpu_inside_drop
        action: drop            --> Drop all other IPv4 traffic
        statistics per-entry
Vlan access-map __epbr_dpu_inside_catchall 3
        match ipv6: __epbr_ipv6_dpu_inside_bypass
        action: forward         --> Regular forwarding for IPv6 multicast traffic
        statistics per-entry
Vlan access-map __epbr_dpu_inside_catchall 4
        match ipv6: __epbr_ipv6_dpu_inside_drop
        action: drop            --> Drop all other IPv6 traffic
        statistics per-entry
```

Step 5 Verify that the VLAN access-map is attached to the FW VLANs.

Example:

```
switch# show vlan filter access-map __epbr_dpu_inside_catchall
vlan map __epbr_dpu_inside_catchall:
        Configured on VLANs:    2003,2005  --> Attached to both FW VLANs
```

Step 6 Verify the ACL counters to detect if any bridged-traffic in the VLANs are matching the access-lists with the drop action.

Example:

```
switch# show vlan access-list __epbr_dpu_inside_catchall
VLAN access-map __epbr_dpu_inside_catchall 1
        IP access list __epbr_ip_dpu_inside_bypass
        2 permit ip any 224.0.0.0 15.255.255.255 [match=0]
        3 permit ip any 255.255.255.255/32 [match=0]
VLAN access-map __epbr_dpu_inside_catchall 2
        IP access list __epbr_ip_dpu_inside_drop
        4294967295 permit ip any any [match=23755974033]  --> Unicast bridged traffic is dropped
VLAN access-map __epbr_dpu_inside_catchall 3
        IPv6 access list __epbr_ipv6_dpu_inside_bypass
        2 permit ipv6 any ff00:: ff:ffff:ffff:ffff:ffff:ffff:ffff:ffff [match=0]
VLAN access-map __epbr_dpu_inside_catchall 4
        IPv6 access list __epbr_ipv6_dpu_inside_drop
        4294967295 permit ipv6 any any [match=12237925993]
```

Step 7 (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 firewall VRFs and VLANs in the out-of-service state by dropping or bypassing it according to the programmed policies, 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.

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

Example:

```
switch(config)# show run service-acceleration | section 'service firewall'
  service firewall
    vrf tenant1 module-affinity dynamic
    vrf tenant2 module-affinity dynamic
    vlan id 2003,2005 bridged-traffic module-affinity dynamic
    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 and VLANs.

Example:

```
switch# show service-acceleration status details
  Service System: hypershield
    Source Interface: loopback2 (10.29.251.4)
    DPU Load-balance Mode: symmetric-hash
    Agent Status: firewall-ready,redirect-installed --> no firewall-agent-connection-pending status
    Agent Health Status: ok
    Controller Connection Status: success --> should successfully connect to controller
    Agent HA Status: not-initialized
    Services:
      Firewall: in-service

      VRF                      Operational State       Affinity(DPU)
      ================================================================
      tenant1                  forwarding ready        1-4
      tenant2                  forwarding ready        1-4

      VLAN                     Operational State       Affinity(DPU)
      ================================================================
      2003                     forwarding ready        1-4
      2005                     forwarding ready        1-4
```

This command provides a detailed status. Verify these:

- The `Agent Status` indicates readiness, for example, `firewall-ready`, `redirect-installed`.
- The `Firewall` service status is `in-service`.
- Each configured VRF and VLAN shows an `Operational State` of `forwarding ready`.
- The `Affinity(DPU)` for each VRF shows a valid DPU number with load-balance mode of pinning or shows all DPUs when symmetric-hash load-balance mode is in use.

After performing these checks, you can 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.

Step 1 Run the Layer 3 troubleshooting command for a specific IPv4 flow.

Example:

```
switch# show troubleshoot l3 ipv4 165.1.1.9 src-ip 166.1.1.9 vrf tenant1
========================================
PROCESSING VRF CONSISTENCY CHECK REQUEST
========================================
VRF 'tenant1' is configured in Service-acceleration
Executing Service-acceleration consistency check for VRF 'tenant1'
**************************************
SERVICE-ACCELERATION CONFIG CONSISTENCY
**************************************
 SUCCESS: Firewall admin state is in-service.
 SUCCESS: VRF 'tenant1' is configured with dynamic module affinity (0)
 SUCCESS: Module affinity matches for VRF 'tenant1' - Configured: 0, Programmed: 1-4
**************************************
SAS OPERATIONAL CONSISTENCY
**************************************
 SUCCESS: VRF 'tenant1' operational state 'forwarding ready' is consistent with admin state 'in-service'
**************************************
SAS REDIRECT CONSISTENCY
**************************************
 SUCCESS: Interface Service-Port-Channel512.13 state check passed
 SUCCESS: Pass-1 ACLs in VRF tenant1 have consistent redirect ports: SPo512.13
 SUCCESS: Pass-2 ACLs in VRF tenant1 have consistent redirect ports: SPo512.13
 SUCCESS: ACL: Interface Vlan656 has 2 redirect rule(s) for VRF tenant1
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-1(__epbr_ip_dpu_inside) on interface Vlan656 in VRF
 tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-1(__epbr_ipv6_dpu_inside) on interface Vlan656 in
VRF tenant1 has valid Redirect interface SPo512.13 in HW
SUCCESS: ACL: Interface Vlan657 has 2 redirect rule(s) for VRF tenant1
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-1(__epbr_ip_dpu_inside) on interface Vlan657 in VRF
 tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-1(__epbr_ipv6_dpu_inside) on interface Vlan657 in
VRF tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: Interface Ethernet1/12.101 has 2 redirect rule(s) for VRF tenant1
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-1(__epbr_ip_dpu_inside) on interface Ethernet1/12.101
 in VRF tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-1(__epbr_ipv6_dpu_inside) on interface Ethernet1/12.101
 in VRF tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: Interface Ethernet1/16/4.101 has 2 redirect rule(s) for VRF tenant1
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-1(__epbr_ip_dpu_inside) on interface Ethernet1/16/4.101
 in VRF tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-1(__epbr_ipv6_dpu_inside) on interface
Ethernet1/16/4.101 in VRF tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: Interface Service-Port-Channel512 has 4 redirect rule(s) for VRF tenant1
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-2(__epbr_ip_dpu_post_fwd) on interface
Service-Port-Channel512 in VRF tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-2(__epbr_ip_dpu_post_fwd) on interface
Service-Port-Channel512 in VRF tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-2(__epbr_ipv6_dpu_post_fwd) on interface
```

```
Service-Port-Channel512 in VRF tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: policy: SVC Redir L3 Policy Pass-2(__epbr_ipv6_dpu_post_fwd) on interface
Service-Port-Channel512 in VRF tenant1 has valid Redirect interface SPo512.13 in HW
 SUCCESS: ACL: Found redirect rules on 5 interface(s) for VRF tenant1
**************************************
POLICY ENFORCEMENT CONSISTENCY
**************************************
 SUCCESS: All data-plane interfaces for vrf tenant1 are present in configured interfaces of Pass-1
ACL '__epbr_ip_dpu_inside'
 SUCCESS: All data-plane interfaces for vrf tenant1 are present in active interfaces for Pass-1 ACL
 '__epbr_ip_dpu_inside'
 SUCCESS: All data-plane interfaces for vrf tenant1 are present in configured interfaces of Pass-1
ACL '__epbr_ipv6_dpu_inside'
 SUCCESS: All data-plane interfaces for vrf tenant1 are present in active interfaces for Pass-1 ACL
 '__epbr_ipv6_dpu_inside'
**********************************
HARDWARE ASIC TYPE: sone
**********************************
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 165.1.1.0/24
```

Specify the destination IP (`165.1.1.9`), source IP (`166.1.1.9`), and VRF name (`tenant1`) for the flow you are troubleshooting.

Analyze the output, which includes:

- Service acceleration consistency checks for the specified VRF.
- Hardware programming consistency checks for the access-lists.
- 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.

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

Step 2 (Optional) Run the Layer 3 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.

Step 3 Run the consistency checker for FW VLAN to detect any bridged traffic redirection issues.

Example:

```
switch# show consistency-checker service-acceleration vlan 2005
========================================
PROCESSING VLAN CONSISTENCY CHECK REQUEST
========================================
VLAN '2005' is configured in Service-acceleration
Executing Service-acceleration consistency check for VLAN '2005'
****************************************
SERVICE-ACCELERATION CONFIG CONSISTENCY
****************************************
 SUCCESS: Firewall admin state is in-service.
****************************************
SAS MODULE AFFINITY CONSISTENCY
****************************************
 SUCCESS: VLAN '2005' is configured with dynamic module affinity (0)
 SUCCESS: Module affinity matches for VLAN '2005' - Configured: 0, Programmed: 1-4
****************************************
SAS OPERATIONAL CONSISTENCY
****************************************
 SUCCESS: VLAN '2005' operational state 'forwarding ready' is consistent with admin state 'in-service'
****************************************
SAS VLAN ACCESS MAP CONSISTENCY
****************************************
 SUCCESS: VLAN access map '__epbr_dpu_inside___dpu_all_dpu_vlan_redir' matches policy
'__dpu_all_dpu_vlan_redir' for VLAN '2005'
****************************************
SAS VLAN FILTER CONSISTENCY
****************************************
 SUCCESS: VLAN filter '__epbr_dpu_inside___dpu_all_dpu_vlan_redir' matches policy
'__dpu_all_dpu_vlan_redir' for VLAN '2005'
****************************************
SAS VLAN TCAM REDIRECT CONSISTENCY
****************************************
 SUCCESS: ACL: Vlan 2005 has 2 redirect rule(s)
 SUCCESS: ACL: policy: SVC Redir L2 Policy Pass-1(__epbr_dpu_inside___dpu_all_dpu_vlan_redir) on vlan
 2005 has valid Redirect interface Service-Port-Channel512 in HW
 SUCCESS: ACL: policy: SVC Redir L2 Policy Pass-2(__epbr_ip_dpu_post_fwd_all) on interface
Service-Port-Channel512 has valid Redirect interface Service-Port-Channel512 in HW
 SUCCESS: ACL: policy: SVC Redir L2 Policy Pass-2(__epbr_ipv6_dpu_post_fwd_all) on interface
Service-Port-Channel512 has valid Redirect interface Service-Port-Channel512 in HW
 SUCCESS: ACL: L2 Pass 2 ACL (Service-Port-Channel512) has 2 redirect rule(s)
```

If the consistency checker command completes successfully for the VLAN, it indicates that the system has the necessary DPU redirection rules configured in place for bridged-traffic in the VLAN.

Hypershield Agent to NX-OS Interactions

Verify that the Hypershield Agent (HSA) has successfully pushed the necessary configuration objects to NX-OS through 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.

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

Example:

```
switch# show system internal service-acceleration dynamic configuration --> agent driven config
isolated from user driven config
<snip>>
vlan 2003
  epbr policy __dpu_all_dpu_vlan_redir
vlan 2005
  epbr policy __dpu_all_dpu_vlan_redir  --> enforce redirection policy on VLAN
vrf context tenant1
  epbr policy __tenant1_dpu_redir       --> enforce redirection policy on VRF
vrf context tenant2
  epbr policy __tenant2_dpu_redir

<<snip>>

epbr service __dpu_all_dpu_vlan_redir type dpu-bridge
 service-end-point module all            --> send to all DPU with symmetric hash for bridged traffic


epbr service __tenant1_dpu_redir type dpu
  vrf tenant1                             --> VRF specific service
  service-end-point module all vlan 13    --> Encap VLAN ID for VRF and send to all DPU with symmetric
 hash

epbr service __tenant2_dpu_redir type dpu
  vrf tenant2
  service-end-point module all vlan 12

<<snip>>

epbr policy __dpu_all_dpu_vlan_redir
  statistics
  match ipv6 address __dpu_ipv6_redir          --> match all v4 traffic
    10 set service __dpu_all_dpu_vlan_redir fail-action drop    --> redirect to service
  match ip address __dpu_redir                 --> match all v6 traffic
    10 set service __dpu_all_dpu_vlan_redir fail-action drop
epbr policy __tenant1_dpu_redir
  statistics
  match ipv6 address __dpu_ipv6_redir          --> match all v4 traffic
    10 set service __tenant1_dpu_redir fail-action drop   --> redirect to service
  match ip address __dpu_redir                 --> match all v6 traffic
    10 set service __tenant1_dpu_redir fail-action drop
epbr policy __tenant2_dpu_redir
  statistics
  match ipv6 address __dpu_ipv6_redir
    10 set service __tenant2_dpu_redir fail-action drop
  match ip address __dpu_redir
    10 set service __tenant2_dpu_redir fail-action drop
```

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

By performing these checks, you can confirm that the Hypershield Agent has successfully pushed the necessary redirection configuration objects to NX-OS using DME, defining how traffic for Firewall VRFs and VLANs should be redirected to the appropriate DPUs.

NPU to DPU Redirection Issues

Verify that the redirection policies are correctly programmed in the NPU hardware (TCAM) and that traffic for the VRFs and VLANs is successfully redirected to the appropriate DPU through service port-channel subinterfaces and service port-channel interfaces. 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.

Step 1 Identify the service port-channel sub-interfaces created for each VRF to redirect the VRF traffic to the DPU and verify their status. Identify the service port-channel in use for each VLAN to redirect the bridged traffic in the VLAN to the DPU and verify the status.

Example:

```
switch# show service-acceleration redirect-policy brief
VRF                              AF Type Interface[Status]  Affinity Redirect Status
================================================================================================
tenant1                          IPv4    SPo512.13[UP]      1-4      Enabled
tenant2                          IPv4    SPo512.12[UP]      1-4      Enabled
tenant1                          IPv6    SPo512.13[UP]      1-4      Enabled
tenant2                          IPv6    SPo512.12[UP]      1-4      Enabled
VLAN                             AF Type Interface[Status]  Affinity Redirect Status
================================================================================================
2003                             IPv4    SPo512[UP]         1-4      Enabled
2005                             IPv4    SPo512[UP]         1-4      Enabled
2003                             IPv6    SPo512[UP]         1-4      Enabled
2005                             IPv6    SPo512[UP]         1-4      Enabled
```

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

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

Example:

```
switch# show interface service-port-channel 512.13 br --> verify intf state and encap
--------------------------------------------------------------------------------------------
Service      VLAN    Type Mode   Status  Reason                             Speed   Protocol
Port-Channel
--------------------------------------------------------------------------------------------
SPo512.13    13      eth  routed up      none                                a-200G(D)    --
```

Specify the sub-interface (`SPO 512.13`) identified in the previous step. Verify that the `VLAN` ID matches the one configured 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 tenant1
  VRF: tenant1
  Policy-map: __tenant1_dpu_redir
  Default Traffic Action: Redirect
```

```
  Match clause:
    ip address(access-lists): __dpu_ipv6_redir
    action: Redirect              --> information matches DME objects
    status(inside): CREATED/ENABLED         status(post_fwd): CREATED/ENABLED --> created/enabled
states
      service __tenant1_dpu_redir, sequence 10, fail-action Drop
        service-module 1-4 vlan 13 [UP] [SPo512.13]    --> correct interface, state, and vlan
  Match clause:
    ip address(access-lists): __dpu_redir
    action: Redirect
    status(inside): CREATED/ENABLED         status(post_fwd): CREATED/ENABLED
      service __tenant1_dpu_redir, sequence 10, fail-action Drop
        service-module 1-4 vlan 13 [UP] [SPo512.13]
```

Replace `tenant1` 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 (`SPo512.13`) information is correct and matches the expected DPU and sub-interface for this VRF.

Step 4 Verify detailed policy programming information in ePBR for bridged traffic in VLAN.

Example:

```
switch# show service-acceleration redirect-policy vlan 2005
  VLAN: 2005
  Policy-map: __dpu_all_dpu_vlan_redir
   Default Traffic Action: Redirect
  Match clause:
    ip address(access-lists): __dpu_redir
    action: Redirect       --> information matches DME objects
    status(inside): CREATED/ENABLED         status(post_fwd): CREATED/DISABLED --> created/enabled
states
      service __dpu_all_dpu_vlan_redir, sequence 10, fail-action Drop
        service-module 1-4 [UP] [SPo512]   --> correct interface, state
  Match clause:
    ip address(access-lists): __dpu_ipv6_redir
    action: Redirect
    status(inside): CREATED/ENABLED         status(post_fwd): CREATED/DISABLED
      service __dpu_all_dpu_vlan_redir, sequence 10, fail-action Drop
        service-module 1-4 [UP] [SPo512]
```

Replace `2005` with the VLAN number. Verify that the `status(inside)` is `CREATED/ENABLED` and `status(post_fwd)` is `CREATED/DISABLED`. Confirm that the `service-module` and service interface (`SPo512`) information is correct and matches the expected DPU for the VLAN.

Step 5 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=0]
        2 permit ip any 224.0.0.0 15.255.255.255 [match=0]
        26201 permit ip any any nve vni 4 redirect Service-Port-Channel512.13 [match=105469965]
--> VNI 4 matches IPv4 table ID for VRF tenant1 mapped to SPO 512.13
        31101 permit ip any any nve vni 5 redirect Service-Port-Channel512.12 [match=105473341]
        4294967295 permit ip any any redirect Null0
```

Example:

```
(IPv6 traffic)
switch# show access-lists  __epbr_ipv6_dpu_inside dynamic
```

```
IPv6 access list __epbr_ipv6_dpu_inside
        statistics per-entry
        1 permit udp any any eq 3785 ttl 255 [match=0]
        2 permit ipv6 any ff00:: ff:ffff:ffff:ffff:ffff:ffff:ffff:ffff [match=0]
        25901 permit ipv6 any any nve vni 4 redirect Service-Port-Channel512.13 [match=56821638]
 --> VNI 4 matches IPv4 table ID for VRF tenant1 mapped to SPO 512.13
        30801 permit ipv6 any any nve vni 5 redirect Service-Port-Channel512.12 [match=56823422]
        4294967295 permit ipv6 any any redirect Null0
```

Identify the rules relevant to the VRF of interest by identifying the table id for the VRF using the **show vrf** *vrf* **detail** command 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 6 Verify that the VLAN access-maps are created and attached to the VLAN for bridged IPv4 or IPv6 traffic.

Example:

```
switch# show vlan filter
vlan map __epbr_dpu_inside___dpu_all_dpu_vlan_redir:
        Configured on VLANs:    2003,2005  --> shared VACL attached to FW VLANs with symmetric hash
switch# show vlan access-map __epbr_dpu_inside___dpu_all_dpu_vlan_redir
Vlan access-map __epbr_dpu_inside___dpu_all_dpu_vlan_redir 1
        match ip: __epbr_ip_dpu_inside_bypass  --> same bypasses for multicast/broadcast
        action: forward
        statistics per-entry
Vlan access-map __epbr_dpu_inside___dpu_all_dpu_vlan_redir 3
        match ipv6: __epbr_ipv6_dpu_inside_bypass
        action: forward
        statistics per-entry
Vlan access-map __epbr_dpu_inside___dpu_all_dpu_vlan_redir 19801
        match ip: __epbr_dpu_inside_ip___dpu_all_dpu_vlan_redir
        action: redirect Service-Port-Channel512     --> Redirect ipv4 traffic to SPO
        statistics per-entry
Vlan access-map __epbr_dpu_inside___dpu_all_dpu_vlan_redir 20101
        match ipv6: __epbr_dpu_inside_ipv6___dpu_all_dpu_vlan_redir
        action: redirect Service-Port-Channel512     --> Redirect ipv6 traffic to SPO
        statistics per-entry
```

Step 7 Verify the access-lists used inside the access-maps and the VACL counters.

Example:

```
switch# show access-lists __epbr_dpu_inside_ip___dpu_all_dpu_vlan_redir  dynamic
IP access list __epbr_dpu_inside_ip___dpu_all_dpu_vlan_redir
        19801 permit ip any any       --> all ipv4 traffic except bypassed traffic
switch# show access-lists __epbr_dpu_inside_ipv6___dpu_all_dpu_vlan_redir  dynamic
IPv6 access list __epbr_dpu_inside_ipv6___dpu_all_dpu_vlan_redir
        20101 permit ipv6 any any     --> all ipv6 traffic except bypassed traffic
```

Step 8 (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                         6       7050    0.08
Ingress SVC Redir L3 Policy Pass-1-IPV6                         6       3522    0.17
Ingress SVC Redir L3 Policy Pass-2-IPV4                         6       7050    0.08
Ingress SVC Redir L3 Policy Pass-2-IPV6                         6       3522    0.17
Ingress SVC Redir L2 Policy Pass-1-IPV4                         5       7051    0.07
Ingress SVC Redir L2 Policy Pass-1-IPV6                         4       3524    0.11
Ingress SVC Redir L2 Policy Pass-2-IPV4                         2       7054    0.02
Ingress SVC Redir L2 Policy Pass-2-IPV6                         2       3526    0.05
Ingress SVC Redir L2 Policy Pass-1-MAC                          0       7040    0.00
Ingress SVC Redir L3 Policy Pass-1-IPV4                         6       7050    0.08
Ingress SVC Redir L3 Policy Pass-1-IPV6                         6       3522    0.17
```

```
Ingress SVC Redir L3 Policy Pass-2-IPV4                         6       7050    0.08
Ingress SVC Redir L3 Policy Pass-2-IPV6                         6       3522    0.17
Ingress SVC Redir L2 Policy Pass-1-IPV4                         5       7051    0.07
Ingress SVC Redir L2 Policy Pass-1-IPV6                         4       3524    0.11
Ingress SVC Redir L2 Policy Pass-2-IPV4                         2       7054    0.02
Ingress SVC Redir L2 Policy Pass-2-IPV6                         2       3526    0.05
Ingress SVC Redir L2 Policy Pass-1-MAC                          0       7040    0.00
Label LBL AO,  Ingress IPv4 SVC Redir L3 Policy Pass-1
      2       125     1.57
Label LBL AP,  Ingress IPv6 SVC Redir L3 Policy Pass-1
      2       125     1.57
Label LBL AQ,  Ingress IPv4 SVC Redir L3 Policy Pass-2
      2       125     1.57
Label LBL AR,  Ingress IPv6 SVC Redir L3 Policy Pass-2
      2       125     1.57
Label LBL AY,  Ingress IPv4 SVC Redir L2 Policy Pass-1 / Ingress IPv6 SVC Redir L2 Policy Pass-1  /
 Ingress MAC SVC Redir L2 Policy Pass-1 2       125     1.57
Label LBL BA,  Ingress IPv4 SVC Redir L2 Policy Pass-2
      2       125     1.57
Label LBL BB,  Ingress IPv6 SVC Redir L2 Policy Pass-2
      2       125     1.57
```

Verify hardware resource usage for ACL policies, including `Pass-1` and `Pass-2` redirect policies and L2 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 firewall VRFs and VLANs to the appropriate DPU through the service port-channel sub-interfaces or service port-channel 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 port-channel 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.

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 learned.

Example:

```
(IPv4 traffic)
switch#  show access-lists __epbr_ip_dpu_post_fwd dynamic
IP access list __epbr_ip_dpu_post_fwd
        statistics per-entry
        26201 permit ip any any nve vni 4 redirect Service-Port-Channel512.13 [match=117821124]
        31101 permit ip any any nve vni 5 redirect Service-Port-Channel512.12 [match=117826975]
        4294967295 permit ip any any [match=0]
```

Example:

```
(IPv6 traffic)
switch# show access-lists __epbr_ipv6_dpu_post_fwd dynamic
IPv6 access list __epbr_ipv6_dpu_post_fwd
        statistics per-entry
        25901 permit ipv6 any any nve vni 4 redirect Service-Port-Channel512.13 [match=60695847]
        30801 permit ipv6 any any nve vni 5 redirect Service-Port-Channel512.12 [match=60698895]
        4294967295 permit ipv6 any any [match=0]
```

Identify the rules relevant to the VRF of interest by identifying the table ID for the VRF using **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 port-channel subinterface for the VRF. Confirm the packet hit counters for new flow learns using these VRF rules.

Step 2 Verify the packet counters matching the internally generated policies for the VLANs to redirect to the DPU for the second pass of inspection before the flows are learned.

Example:

```
switch# show access-lists __epbr_ip_dpu_post_fwd_all dynamic

IP access list __epbr_ip_dpu_post_fwd_all
        statistics per-entry
        4294967295 permit ip any any redirect Service-Port-Channel 512 [match=0] --> resend to DPU
for Pass2
switch# show access-lists __epbr_ipv6_dpu_post_fwd_all dynamic

IPv6 access list __epbr_ipv6_dpu_post_fwd_all
        statistics per-entry
        4294967295 permit ipv6 any any redirect Service-Port-Channel 512 [match=0]
```

For pinning mode unique ACLs per DPU are created.

Step 3 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 interface service-port-channel 512
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-port channel 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.

These 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

Step 1 Capture packets received on the ingressing Layer 3 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)# 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 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 DPU decap happens, but you can get insights into the received DPU header. For routed-traffic, the service-vlan to filter on can be obtained from the VLAN information in **show service-acceleration redirect-policy vrf** *name*. For bridged-traffic use the forwarding VLAN of the packet as the service VLAN.

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 outer ipv4 src-ip 10.1.0.2 dst-ip 16.1.0.2
```

```
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, specifically 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 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 The snapshot is taken before the DPU header encapsulation happens, so the 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.

Inter-VRF Traffic Inspection

When troubleshooting Inter-VRF traffic inspection using route leak details, the route is initially learned in the source VRF, for example, red, and then leaked to the target VRF, for example, green, as shown by route entries with next-hop information referencing the source VRF.

Running the **show troubleshoot** command provides a comprehensive view of the route leak. The output displays route information from the PI RIB, FIB, and hardware layers. This confirms the route's presence and next-hop details across VRFs and verifies its installation in hardware for proper forwarding. This layered inspection helps identify where the route leak is functioning correctly or where issues can arise in the Inter-VRF traffic path.

Use this procedure to verify the route leak details for Inter-VRF traffic inspection and troubleshoot accordingly.

Step 1 Use the **show ip route** *ip-address* **vrf** *vrf-name* command for the source VRF, for example, VRF red.

Example:

```
show ip route 12.10.1.0/24 vrf red
IP Route Table for VRF "red"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
```

```
'%<string>' in via output denotes VRF <string>

12.10.1.0/24, ubest/mbest: 1/0
*via 1.10.1.2, [1/0], 00:02:58, static
```

Step 2 Use the **show ip route** *ip-address* **vrf** *vrf-name* command for the target VRF, for example, VRF green.

Example:

```
show ip route 12.10.1.0/24 vrf green
IP Route Table for VRF "green"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

12.10.1.0/24, ubest/mbest: 1/0
*via 1.10.1.2%red, [20/0], 00:13:58, bgp-100, external, tag 100
```

Step 3 Use the **troubleshoot** command to see the route leak details. The route details are shown from URIB, FIB, and the hardware.

Example:

```
show troubleshoot l3 ipv4 12.10.1.0/24 vrf green
************************************
1. CHECK ROUTE IN PI RIB
************************************
show ip route 12.10.1.0/24 vrf green
prefix : 12.10.1.0
mask length : 24
IP Route Table for VRF "green"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
12.10.1.0/24, ubest/mbest: 1/0
*via 1.10.1.2%red, [20/0], 00:10:15, bgp-100, external, tag 100
************************************
3. CHECK HOST ROUTE IN FIB AND HARDWARE
************************************
show forwarding route 12.10.1.0/24 platform vrf green module 1
Prefix 12.10.1.0/24, No of paths: 1, Update time: Fri May 1 04:37:41 2026
SGT: 0
DPI: 0
Vobj id: 1 Partial Install: No
1.10.1.2 Ethernet1/10
HH:0x80b78700 Flags:0x0 Extended Flags:0x100 Holder:0x1 Next_obj_type:9
Hw-idx vobj: unit-0:0x40001 unit-1:0x0 unit-2:0x0, cmn-index: 500009, LIF:16394 buf-idx 0(0)
************************************
4. CHECK ROUTE,NEXTHOP IN HARDWARE
************************************
dchal module 1 "route show 0 4 12.10.1.0 24"
IP Address: b'12.10.1.0'
Length: 24
Unit: 0
VRF: 4
VRF Redirect Id: 0x6
```

High Availability

Use this procedure to verify the details of High Availability (HA) and troubleshoot issues for these scenarios accordingly:

- connectivity issues
- both peers are in `no in-service` mode
- one peer is in-service mode and the other is not
- both peers are in in-service mode
- membership failures

High Availability connectivity issues

High Availability (HA) connectivity between Hypershield Agents is accomplished only when:

- HA configuration is complete, correct, and symmetric
- HA is enabled, and
- Service Firewall is configured.

Troubleshooting connectivity issues involves these steps:

- Verify the HA configuration.
- Verify the network connectivity between the peers and the containers.
- Verify Hypershield Agent states for High Availability.

Step 1 Verify the High Availability status.

Example:

```
show service-acceleration high-availability status
  Service System: hypershield
    HA Source Interface: loopback1 (192.1.1.1)
    HA Admin State: no-shutdown
    Agent Status for Service Firewall: not-ready
    Agent HA Status: not-ready
    Peers:
      PeerIP              Peer Service State  HA State with Peer  Reason
      ------------------- ------------------  ------------------  --------------
      192.1.1.2           unknown             no-ha               no peer connectivity
```

Step 2 Verify Service-acceleration configuration and network configuration and routes on each peer.

- HA config is complete, correct and symmetric
- HA is enabled
- Service Firewall is configured.

Example:

```
switch# show run service-acceleration
<<snip>>
version 10.6(3q) Bios:version 01.17
feature service-acceleration
service system hypershield
  source-interface loopback2
  high-availability
    source-interface loopback1
    peer 192.1.1.2
```

```
    no shutdown
  service firewall
    vrf tenant1 module-affinity dynamic
    vrf tenant2 module-affinity dynamic
    vlan id 2003,2005 bridged-traffic module-affinity dynamic
```

Step 3 Verify the networking configuration has been set up for the HA connectivity to the configured peer from the local HA IP. In this example, the loopback interface is configured with the correct IP address and OSPF settings, and the routing table shows the route to the destination reachable using OSPF through the Ethernet interface.

Example:

```
switch# show run interface loopback 1
<<snip>>
version 10.6(3q) Bios:version 01.17
interface loopback1
  ip address 192.1.1.1/32
  ip ospf cost 1
  ip router ospf 10 area 0.0.0.0
  ipv6 router ospfv3 10 area 0.0.0.0
switch# show ip route 192.1.1.2
IP Route Table for VRF "default"
<<snip>>
192.1.1.2/32, ubest/mbest: 1/0
    *via 192.1.1.2, Eth1/16/3, [110/2], 00:22:46, ospf-10, intra
     via 192.1.1.2, Eth1/16/3, [250/0], 00:20:51, am
```

Step 4 Run HA consistency checker on each peer to validate the NX-OS configuration with the Hypershield Agent driven states and detect discrepancies.

Example:

```
switch# show consistency-checker service-acceleration high-availability
============================================================
PROCESSING HIGH-AVAILABILITY CONSISTENCY CHECK REQUEST
============================================================
**************************************
HA FIREWALL STATE CONSISTENCY
**************************************
  [PASS] Firewall configured and internal states match: out-of-service --> Firewall is created
  [PASS] Agent's firewall readiness state matches internal readiness state: not-ready
  [PASS] Agent's firewall readiness state matches published readiness state: not-ready
**************************************
FW VLAN OPERATIONAL STATE CONSISTENCY
**************************************
FW L2, L3 VLAN Set Consistency Checks
======================================
  Expected L3 FW VLANs (from VRFs): 656-657,659,755,859
  Expected L2 FW VLANs : 2003,2005
  Expected Combined L2+L3 VLANs: 656-657,659,755,859,2003,2005
  L3 VLAN Comparison:
    [PASS] L3 VLANs match published FW L3 VLANs
  L2 VLAN Comparison:
    [PASS] L2 VLANs match published FW L2 VLANs
  Combined L2+L3 VLAN Comparison:
    [PASS] Combined L2+L3 FW VLANs identified as expected
**************************************
HA PEER CONSISTENCY
**************************************
HA Configuration and Peer State Checks
======================================
  HA Admin State Comparison:
    [PASS] HA admin state matches with internal state: no-shutdown  --> HA is enabled
  HA Source interface Comparison:
```

```
    [PASS] HA Source interface matches: loopback1
  HA Source IP Address Comparison:
    [PASS] HA source IP address matches: 192.1.1.2
  HA Peer Configuration Validation:
    [PASS] Peer 192.1.1.1 found internally.
    HA Peer State Validation:
    [PASS] Internal Peer service state for peer 192.1.1.1 matches service state published by agent:
 unknown
            --> Peer is configured
    [PASS] Peer 192.1.1.1 derived firewall state matches: 'unknown'
  [NOTE] HA Troubleshooting Suggestion:
    Detected 1 peer(s) with service state unknown:
      - Peer 192.1.1.1: unknown
    To identify HA peer connectivity issues, run:
      service system hypershield test high-availability
HA VPC Peer Consistency Checks
==============================
  Validating HA Peer Info:
  Internal Peer state validation:
    VPC peer is alive
    [PASS] Peer 192.1.1.1 is being detected as an alive VPC peer
  VPC Peer Firewall State Validation:
    [PASS] Internally known HA VPC peer firewall state is 'unknown' (VPC peer exists)
    [PASS] Published HA VPC peer firewall state is 'unknown' (VPC peer exists)
VPC State Consistency Checks
============================
  Validating 4 VPC(s)
  VPC 60 (Po60):
    [Pass] Internal details for VPC 60 matches configuration
    [PASS] VPC 60 is of interest and allows 3 FW l2/l3 VLAN(s): 657,659,2005
  VPC 61 (Po61):
    [Pass] Internal details for VPC 61 matches configuration
    [PASS] VPC 61 is of interest and allows 2 FW l2/l3 VLAN(s): 656,2005
  VPC 62 (Po62):
    [Pass] Internal details for VPC 62 matches configuration
    [PASS] VPC 62 is of interest and allows 2 FW l2/l3 VLAN(s): 755,2003
  VPC 63 (Po63):
    [Pass] Internal details for VPC 63 matches configuration
    [PASS] VPC 63 is of interest and allows 1 FW l2/l3 VLAN(s): 2003
**************************************
HA VRF/VLAN OPERATIONAL STATE CONSISTENCY
    [PASS] 2 VRF(s) were validated and are in 'isolated' state
    [PASS] VLANs 2003,2005 were validated and are in 'VLAN set to drop' state
    [SKIP] VRF/VLAN vpc peer consistency checks skipped - one or both firewalls not ready
    [SKIP] VLAN peer consistency checks skipped - one or both firewalls not ready
================================================================================
HIGH-AVAILABILITY CONSISTENCY CHECK COMPLETED - ALL CHECKS PASSED
================================================================================
```

Step 5 Run the network connectivity troubleshooting tool to detect network connectivity issues to the peer's Hypershield Agent.

Example:

```
switch# service system hypershield test high-availability
==============================
Running HA config checks
==============================
Using loopback 1 as the HA source-interface.
1 HA peers configured.
HA admin state is enabled.
================================================
Running HA operational checks
================================================
local svc is not fully functional : criteria not met: in_service
Hypershield peer 192.1.1.1 not functional: print peer HA svc state reason.
```

```
================================================
Starting network connectivity checks for the peers
================================================
Using curl to check connection to 192.1.1.1 port 28416
curl: (1) Received HTTP/0.9 when not allowed
Curl successfully connected to 192.1.1.1:28416. Collect 'show tech service-acceleration' to debug
any agent connectivity failure
HA network connectivity check completed for peer with IP 192.1.1.1
```

Both peers are in no in-service mode

When both peers are in `no in-service` traffic deflection for vPCs and bridged traffic is turned off in both vPC peers, vPCs carrying VLANs that are Layer 3 or Layer 2 firewall VLANs are not suspended.

VRFs remain in isolated state on both peers. For more information, see Network configuration before firewall bringup, on page 8.

Any traffic arriving for VRFs or VLANs drops. For more information, see Network configuration before firewall bringup, on page 8.

Note

DPU keepalives and syncs are disabled when firewall is not in-service. Only Hypershield Agents establish HA connectivity at this time.

Step 1 Both peers must report local state as not-ready and peer service state as not-ready. Peer HA state is unavailable on both sides.

Example:

```
switch(config)# show service-acceleration high-availability status
  Service System: hypershield
    HA Source Interface: loopback1 (192.1.1.1)
    HA Admin State: no-shutdown
    Agent Status for Service Firewall: not-ready   --> local service is not ready
    Agent HA Status: unavailable
    Peers:
      PeerIP              Peer Service State  HA State with Peer  Reason
      ------------------- ------------------  ------------------  --------------
      192.1.1.2           not-ready           ha-unavailable      local and peer service failure
```

Step 2 Verify Hypershield Agent HA information/configuration.

Example:

```
switch # service system hypershield connect session /usr/src/app/agwctl ha info
Client got: ok
Data:
=== HA Info ===
Admin State:        enabled
Oper State:         ha-ready
Leader:            false  --> After HA connectivity is established peer with lower IP/ready is leader
HA IP:              192.1.1.2
HA Port:            28416
Policy Check:
Policy Rev:
Configured Peers:   192.1.1.1
<<snip>>
```

Example:

Output on the HA peer

```
switch(config)# service system hypershield connect session /usr/src/app/agwctl ha info
Client got: ok
Data:
=== HA Info ===
Admin State:        enabled
Oper State:         ha-ready
Leader:             true    --> After HA connectivity is established peer with lower IP/ready is
leader
HA IP:              192.1.1.1
HA Port:            28416
Policy Check:
Policy Rev:
Configured Peers:   192.1.1.2
<<snip>>
```

One peer is in in-service mode and the other is not

When one peer is in `no in-service` and other peer is in `in-service`, traffic must be deflected towards peer that is in-service.

- Verify that the FW service in Hypershield Agent and NXOS is ready on the peer that is `in-service`.
- Verify state visibility between the peers.
- Verify VPC and bridged traffic deflection has been turned off on the peer that is FW ready.
- Verify VRFs are no longer isolated and traffic is able to redirect to DPU on the peer that is FW ready. For more information, see Packet Forwarding, on page 14.
- Verify vPC and bridged traffic deflection is in effect on the peer that is FW not-ready.
- DPU keepalives and syncing are still disabled on the peer that is not in-service.

Step 1 Verify the HA status.

Example:

Output on the peer that is not ready

```
switch# show service-acceleration high-availability status
  Service System: hypershield
    HA Source Interface: loopback1 (192.1.1.1)
    HA Admin State: no-shutdown
    Agent Status for Service Firewall: not-ready
    Agent HA Status: unavailable
    Peers:
      PeerIP         Peer Service State  HA State with Peer  Reason
      -------------- ------------------  ------------------  --------------
      192.1.1.2      ready               ha-unavailable      local service failure
```

Example:

Output on the peer that is ready

```
switch# show service-acceleration high-availability status
  Service System: hypershield
    HA Source Interface: loopback1 (192.1.1.2)
    HA Admin State: no-shutdown
    Agent Status for Service Firewall: ready
    Agent HA Status: not-ready
    Peers:
      PeerIP         Peer Service State  HA State with Peer  Reason
```

```
-------------- ------------------  ------------------  --------------
192.1.1.1      not-ready           ha-unavailable      peer service failure
```

The peer that is in-service reports the local state as ready and remote state as not-ready. Similarly, the peer that is not in-service reports the local state as not-ready and remote state as ready.

The HA state is not-ready on the peer that is onboarded and is unavailable on the other peer.

Step 2 Run HA consistency checker on each peer to validate NX configuration, states with Agent driven states and verify traffic deflection expectations.

Example:

Output on the peer that is not ready

```
switch# show consistency-checker service-acceleration high-availability
=====================================================
PROCESSING HIGH-AVAILABILITY CONSISTENCY CHECK REQUEST
=====================================================
**************************************
HA FIREWALL STATE CONSISTENCY
**************************************
  [PASS] Firewall configured and internal states match: out-of-service
                 --> HA Peer that is not in-service
  [PASS] Agent's firewall readiness state matches internal readiness state: not-ready
  [PASS] Agent's firewall readiness state matches published readiness state: not-ready
**************************************
FW VLAN OPERATIONAL STATE CONSISTENCY
**************************************
FW L2, L3 VLAN Set Consistency Checks
====================================
  Expected L3 FW VLANs (from VRFs): 656-657,659,755,859
  Expected L2 FW VLANs : 2003,2005
  Expected Combined L2+L3 VLANs: 656-657,659,755,859,2003,2005
  L3 VLAN Comparison:
    [PASS] L3 VLANs match published FW L3 VLANs
  L2 VLAN Comparison:
    [PASS] L2 VLANs match published FW L2 VLANs
  Combined L2+L3 VLAN Comparison:
    [PASS] Combined L2+L3 FW VLANs identified as expected
**************************************
HA PEER CONSISTENCY
**************************************
HA Configuration and Peer State Checks
======================================
  HA Admin State Comparison:
    [PASS] HA admin state matches with internal state: no-shutdown
  HA Source interface Comparison:
    [PASS] HA Source interface matches: loopback1
  HA Source IP Address Comparison:
    [PASS] HA source IP address matches: 192.1.1.1
  HA Peer Configuration Validation:
    [PASS] Peer 192.1.1.2 found internally.
  HA Peer State Validation:
    [PASS] Internal Peer service state for peer 192.1.1.2 matches service state published by agent:
 ready
    [PASS] Peer 192.1.1.2 derived firewall state matches: 'ready'
HA VPC Peer Consistency Checks
==============================
  Validating HA Peer Info:
  Internal Peer state validation:
    VPC peer is alive
    [PASS] Peer 192.1.1.2 is being detected as an alive VPC peer --> vPC peer is being detected
  VPC Peer Firewall State Validation:
    [PASS] Internally known HA VPC peer firewall state is 'ready' (VPC peer exists)
```

```
    [PASS] Published HA VPC peer firewall state is 'ready' (VPC peer exists)
VPC State Consistency Checks
============================
  Validating 4 VPC(s)
  VPC 60 (Po60):
    [Pass] Internal details for VPC 60 matches configuration
    [PASS] VPC 60 is of interest and allows 3 FW l2/l3 VLAN(s): 657,659,2005
    [PASS] VPC 60 state is 'down' (local FW not-ready, peer FW ready)
      State reason: suspended by service-acceleration
  VPC 61 (Po61):
    [Pass] Internal details for VPC 61 matches configuration
    [PASS] VPC 61 is of interest and allows 2 FW l2/l3 VLAN(s): 656,2005
    [PASS] VPC 61 state is 'down' (local FW not-ready, peer FW ready)
      State reason: suspended by service-acceleration
  VPC 62 (Po62):
    [Pass] Internal details for VPC 62 matches configuration
    [PASS] VPC 62 is of interest and allows 2 FW l2/l3 VLAN(s): 755,2003
    [PASS] VPC 62 state is 'down' (local FW not-ready, peer FW ready)
      State reason: suspended by service-acceleration
  VPC 63 (Po63):
    [Pass] Internal details for VPC 63 matches configuration
    [PASS] VPC 63 is of interest and allows 1 FW l2/l3 VLAN(s): 2003
    [PASS] VPC 63 state is 'down' (local FW not-ready, peer FW ready)
      State reason: suspended by service-acceleration
**************************************
HA VRF/VLAN OPERATIONAL STATE CONSISTENCY
**************************************
    [PASS] 2 VRF(s) were validated and are in 'isolated' state
    [PASS] VLANs 2003,2005 were validated and are in 'VLAN set to drop' state
    [SKIP] VRF/VLAN vpc peer consistency checks skipped - one or both firewalls not ready
    [SKIP] VLAN peer consistency checks skipped - one or both firewalls not ready
================================================================================================
HIGH-AVAILABILITY CONSISTENCY CHECK COMPLETED - ALL CHECKS PASSED
================================================================================================
```

Step 3 Verify that the vPCs are not suspended on the peer that is not in-service and are not suspended on the peer that is in in-service mode.

Example:

Output on the peer that is not ready

```
switch# show vpc brief
<<snip>>
vPC domain id      : 1
Peer status        : peer adjacency formed ok
vPC keep-alive status    : peer is alive
vPC role                         : primary
<<snip>>
vPC Peer-link status
---------------------------------------------------
id    Port   Status Active vlans
--    ----   ------ --------------------------
1 Po900  up        656-657,659,755,757-
                    759,859,2003,2005,2007,2009
vPC status
---------------------------------------------------
Id    Port          Status Consistency Reason                Active vlans
--    ------------  ------ ----------- ------                ---------------
60    Po60          down*  success     success               -

61 Po61             down*  success     success            -
<<snip>>
switch# show interface port-channel 60-61 br
--------------------------------------------------------------------------------------------
```

```
Port-channel VLAN    Type Mode   Status  Reason                               Speed   Protocol
Interface
--------------------------------------------------------------------------------------------
Po60         1       eth  trunk  down    suspended by service-acceleration     auto(D)  none
Po61         1       eth  trunk  down    suspended by service-acceleration     auto(D)  none
```

Step 4 Verify that the HSRP groups transition to active on the peer that is now ready, as the VRFs are no longer isolated while the HSRP groups remain in INIT state on the peer that is not-ready. As vPC uses HSRP active-active, traffic deflection to the VIP occurs as a result of vPCs being suspended on the not-ready HA vPC peer.

Example:

Output on the peer that is not ready

```
switch# show hsrp group 656
Vlan656 - Group 656 (HSRP-V2) (IPv4)
  Local state is Initial(VRF Isolation Enabled), priority 110 (Cfged 110), may preempt
    Forwarding threshold(for vPC), lower: 0 upper: 110
  Hellotime 3 sec, holdtime 10 sec
  Virtual IP address is 166.1.1.6 (Cfged)
  Active router is unknown
  Standby router is unknown
  Authentication text "cisco"
  Virtual mac address is 0000.abcd.0656 (Cfged MAC)
  3 state changes, last state change 1d14h
  IP redundancy name is hsrp-Vlan656-656 (default)
```

Example:

Output on the peer that is ready

```
switch# show hsrp group 656
Vlan656 - Group 656 (HSRP-V2) (IPv4)
  Local state is Active, priority 100 (Cfged 100), may preempt
    Forwarding threshold(for vPC), lower: 0 upper: 100
  Hellotime 3 sec, holdtime 10 sec
  Next hello sent in 1.394000 sec(s)
  Virtual IP address is 166.1.1.6 (Cfged)
  Active router is local
  Standby router is unknown
  Authentication text "cisco"
  Virtual mac address is 0000.abcd.0656 (Cfged MAC)
  7 state changes, last state change 01:33:43
  IP redundancy name is hsrp-Vlan656-656 (default)
```

Step 5 Verify Hypershield Agent HA information/configuration. The peer that is ready assumes the role of the leader.

Example:

```
switch# service system hypershield connect session /usr/src/app/agwctl ha info
Client got: ok
Data:
=== HA Info ===
Admin State:        enabled
Oper State:         ha-ready
Leader:             true            --> Role switches to leader since it is "ready"
HA IP:              192.1.1.2
HA Port:            28416
Policy Check:       false
Policy Rev:
Configured Peers:   192.1.1.1

switch# service system hypershield connect session /usr/src/app/agwctl ha info
Client got: ok
Data:
=== HA Info ===
```

```
Admin State:        enabled
Oper State:         ha-ready
Leader:             false        --> Role is follower since it is "not-ready"
HA IP:              192.1.1.1
HA Port:            28416
Policy Check:       false
Policy Rev:
Configured Peers:   192.1.1.2
```

Both peers are in in-service mode

When both peers are in in-service mode, traffic deflection must be turned off, when both peers are FW ready:

- Verify the FW readiness on each peer.
- Verify the state visibility between the peers.
- Verify that the vPC and bridged traffic deflection has been turned off on both peers (See previous sections).
- Verify that the VRFs are no longer isolated and traffic is able to redirect to DPU on the peer that is FW ready (Refer to Packet Forwarding, on page 14).
- Verify flow-sync between peers.

Step 1 Verify that both peers report local and peer state as ready. HA state is ready on both peers.

Example:

Output on one peer

```
switch# show service-acceleration high-availability status
  Service System: hypershield
    HA Source Interface: loopback1 (192.1.1.1)
    HA Admin State: no-shutdown
    Agent Status for Service Firewall: ready
    Agent HA Status: ready
    Peers:
      PeerIP         Peer Service State  HA State with Peer  Reason
      -------------- ------------------  ------------------  --------------
      192.1.1.2      ready               ha-ok               all criteria met
```

Example:

Output on the HA peer

```
switch# show service-acceleration high-availability status
  Service System: hypershield
    HA Source Interface: loopback1 (192.1.1.2)
    HA Admin State: no-shutdown
    Agent Status for Service Firewall: ready
    Agent HA Status: ready
    Peers:
      PeerIP         Peer Service State  HA State with Peer  Reason
      -------------- ------------------  ------------------  --------------
      192.1.1.1      ready               ha-ok               all criteria met
```

Step 2 Run HA consistency checker on each peer to validate NX configuration, states with Agent driven states and verify traffic deflection expectations.

Example:

```
switch# show consistency-checker service-acceleration high-availability
=====================================================
```

```
PROCESSING HIGH-AVAILABILITY CONSISTENCY CHECK REQUEST
======================================================
**************************************
HA FIREWALL STATE CONSISTENCY
**************************************
  [PASS] Firewall configured and internal states match: in-service
  [PASS] Agent's firewall readiness state matches internal readiness state: ready
  [PASS] Agent's firewall readiness state matches published readiness state: ready
**************************************
FW VLAN OPERATIONAL STATE CONSISTENCY
**************************************
FW L2, L3 VLAN Set Consistency Checks
=====================================
  Expected L3 FW VLANs (from VRFs): 656-657,659,755,859
  Expected L2 FW VLANs : 2003,2005
  Expected Combined L2+L3 VLANs: 656-657,659,755,859,2003,2005
  L3 VLAN Comparison:
    [PASS] L3 VLANs match published FW L3 VLANs
  L2 VLAN Comparison:
    [PASS] L2 VLANs match published FW L2 VLANs
  Combined L2+L3 VLAN Comparison:
    [PASS] Combined L2+L3 FW VLANs identified as expected
**************************************
HA PEER CONSISTENCY
**************************************
HA Configuration and Peer State Checks
======================================
  HA Admin State Comparison:
    [PASS] HA admin state matches with internal state: no-shutdown
  HA Source interface Comparison:
    [PASS] HA Source interface matches: loopback1
  HA Source IP Address Comparison:
    [PASS] HA source IP address matches: 192.1.1.1
  HA Peer Configuration Validation:
    [PASS] Peer 192.1.1.2 found internally.
  HA Peer State Validation:
    [PASS] Internal Peer service state for peer 192.1.1.2 matches service state published by agent:
 ready
    [PASS] Peer 192.1.1.2 derived firewall state matches: 'ready'
HA VPC Peer Consistency Checks
==============================
  Validating HA Peer Info:
  Internal Peer state validation:
    VPC peer is alive
    [PASS] Peer 192.1.1.2 is being detected as an alive VPC peer
  VPC Peer Firewall State Validation:
    [PASS] Internally known HA VPC peer firewall state is 'ready' (VPC peer exists)
    [PASS] Published HA VPC peer firewall state is 'ready' (VPC peer exists)
VPC State Consistency Checks
============================
  Validating 4 VPC(s)
  VPC 60 (Po60):
    [Pass] Internal details for VPC 60 matches configuration
    [PASS] VPC 60 is of interest and allows 3 FW l2/l3 VLAN(s): 657,659,2005
  VPC 61 (Po61):
    [Pass] Internal details for VPC 61 matches configuration
    [PASS] VPC 61 is of interest and allows 2 FW l2/l3 VLAN(s): 656,2005
  VPC 62 (Po62):
    [Pass] Internal details for VPC 62 matches configuration
    [PASS] VPC 62 is of interest and allows 2 FW l2/l3 VLAN(s): 755,2003
  VPC 63 (Po63):
    [Pass] Internal details for VPC 63 matches configuration
    [PASS] VPC 63 is of interest and allows 1 FW l2/l3 VLAN(s): 2003
**************************************
```

```
HA VRF/VLAN OPERATIONAL STATE CONSISTENCY
****************************************
    [PASS] 2 VRF(s) were validated and are in 'forwarding ready/redirection installed' state
    [PASS] VLANs 2003,2005 were validated and are in 'forwarding ready' state
    [INFO] Both firewalls are ready - proceeding with VRF and VRF-ID validation
    [PASS] All VRF ID validations passed
    [PASS] All VLAN peer consistency checks passed - 2 VLAN(s) validated
================================================================================================
HIGH-AVAILABILITY CONSISTENCY CHECK COMPLETED - ALL CHECKS PASSED
================================================================================================
```

Step 3 The encapsulation VLANs used for the FW VRFs must be consistent on both peers. If these are not consistent, these should be flagged by the CC. These IDs must be consistent on the DPUs of the two switches and may be verified by comparing them to the values visible on NXOS using the **show service-acceleration redirect-policy brief** command.

Example:

```
switch# slot 1 dpu 1 dpctl hs vrf show-map
VRF Table:
VRF Name                        VRF ID     Internal VRF ID
------------------------------------------------------------
tenant1                         12         4
default                         1          1
tenant2                         13         3
Total VRFs: 3
```

Step 4 Verify the Hypershield Agent HA information/configuration.

Example:

Output of the Hypershield Agent HA status on the leader

```
switch# service system hypershield connect session /usr/src/app/agwctl ha show
Client got: ok
Data:
=== HA Status ===
Cluster State:   active/active
NX State:        ha-ready
--- Local ---
  Svc State:         ready since 2026-05-12 20:15:48 UTC
    Reason:          all criteria met
    Criteria:        [OK] since 2026-05-12 20:15:48 UTC
      dpu_healthy:   [OK]
      dpu_insync:    [OK]
      in_service:    [OK]
--- Peer: 192.1.1.1 ---
  HA State:               ha-ok since 2026-05-12 22:35:41 UTC
    Reason:               all criteria met
  Svc State:              ready since 2026-05-12 22:35:40 UTC
    Reason:               peer service ready
    Criteria:             [OK] since 2026-05-12 22:35:40 UTC
      peer_service:       [OK]
  Membership:             [OK] since 2026-05-12 23:52:45 UTC
    peer_compatible:      [OK]
    peer_dpu_keepalive:   [OK]
    peer_vrf_gid:         [OK]
  Adjacency:              [OK] since 2026-05-12 22:35:10 UTC
    peer_dpu_bulk_sync:   [OK]
    peer_policy:          [OK]
```

Example:

Output of the Hypershield Agent HA peer status on the leader that is ready

```
switch# service system hypershield connect session /usr/src/app/agwctl ha peers show
Client got: ok
```

```
Data:
=== HA Peers ===
--- Peer: 192.1.1.1 ---
HA State:        ha-ok since 2026-05-12 22:35:41 UTC
  Reason:        all criteria met
Svc State:       ready since 2026-05-12 22:35:40 UTC
  Reason:        peer service ready
IpConfigState:   success
Connected:       [OK] since 2026-05-12 23:57:40 UTC
Service Criteria Met:   [OK] since 2026-05-12 22:35:40 UTC
  peer_service:         [OK]
Member Criteria Met:    [OK] since 2026-05-12 23:57:44 UTC
  peer_compatible:      [OK]
  peer_dpu_keepalive:   [OK]
  peer_vrf_gid:         [OK]
Adjacency Criteria Met:   [OK] since 2026-05-12 22:35:10 UTC
  peer_dpu_bulk_sync:     [OK]
  peer_policy:            [OK]
Member Info:
  Serial:              FDO28370FGQ
  Model:               N9324C-SE1U
  SW Version:          10.6(3q)ISQ9(0.13)
  CPA Version:         v1.19.0-pre.8-151-g14b22c12
  Reported HA State:   ha-ready
  Service:             ready
  LB Mode:             symmetric-hash
  Policy Rev:
  Policy Check:        false
  DPUs:                dpu2(1.150.2), dpu4(1.150.2), dpu1(1.150.2), dpu3(1.150.2)
DPU HA Status:
  dpu1:   keepalive=[OK]  bulk_sync_local=[OK]  bulk_sync_peer=[OK]
  dpu2:   keepalive=[OK]  bulk_sync_local=[OK]  bulk_sync_peer=[OK]
  dpu3:   keepalive=[OK]  bulk_sync_local=[OK]  bulk_sync_peer=[OK]
  dpu4:   keepalive=[OK]  bulk_sync_local=[OK]  bulk_sync_peer=[OK]
```

Step 5 Verify the Hypershield Agent HA status on the follower that is now ready.

Example:

```
switch# service system hypershield connect session /usr/src/app/agwctl ha show
Client got: ok
Data:
=== HA Status ===
Cluster State:   active/active
NX State:        ha-ready
--- Local ---
  Svc State:         ready since 2026-05-12 20:15:48 UTC
    Reason:          all criteria met
    Criteria:        [OK] since 2026-05-12 20:15:48 UTC
      dpu_healthy:   [OK]
      dpu_insync:    [OK]
      in_service:    [OK]
--- Peer: 192.1.1.1 ---
  HA State:              ha-ok since 2026-05-12 22:35:41 UTC
    Reason:              all criteria met
  Svc State:             ready since 2026-05-12 22:35:40 UTC
    Reason:              peer service ready
    Criteria:            [OK] since 2026-05-12 22:35:40 UTC
      peer_service:      [OK]
  Membership:            [OK] since 2026-05-12 23:52:45 UTC
    peer_compatible:     [OK]
    peer_dpu_keepalive:  [OK]
    peer_vrf_gid:        [OK]
  Adjacency:             [OK] since 2026-05-12 22:35:10 UTC
```

```
    peer_dpu_bulk_sync:   [OK]
    peer_policy:          [OK]
```

Step 6 Verify the Hypershield Agent HA peer status on peer that is ready.

Example:

```
switch3# service system hypershield connect session /usr/src/app/agwctl ha peers show
Client got: ok
Data:
=== HA Peers ===
--- Peer: 192.1.1.2 ---
HA State:        ha-ok since 2026-05-12 22:35:20 UTC
  Reason:        all criteria met  --> Both peers are ready and sync is completed – HA is ready
Svc State:       ready since 2026-05-12 20:15:28 UTC
  Reason:        peer service ready
IpConfigState:   success
Connected:       [OK] since 2026-05-12 23:55:54 UTC
Service Criteria Met:   [OK] since 2026-05-12 20:15:28 UTC
  peer_service:         [OK]
Member Criteria Met:    [OK] since 2026-05-12 23:55:55 UTC
  peer_compatible:      [OK]
  peer_dpu_keepalive:   [OK]
  peer_vrf_gid:         [OK]
Adjacency Criteria Met:   [OK] since 2026-05-12 22:34:50 UTC
  peer_dpu_bulk_sync:     [OK]
  peer_policy:            [OK]
Member Info:
  Serial:              FDO28370FGZ
  Model:               N9324C-SE1U
  SW Version:          10.6(3q)ISQ9(0.13)
  CPA Version:         v1.19.0-pre.8-151-g14b22c12
  Reported HA State:   ha-ready
  Service:             ready
  LB Mode:             symmetric-hash
  Policy Rev:
  Policy Check:        false
  DPUs:                dpu2(1.150.2), dpu4(1.150.2), dpu1(1.150.2), dpu3(1.150.2)
DPU HA Status:
  dpu1:   keepalive=[OK]  bulk_sync_local=[OK]  bulk_sync_peer=[OK]
  dpu2:   keepalive=[OK]  bulk_sync_local=[OK]  bulk_sync_peer=[OK]
  dpu3:   keepalive=[OK]  bulk_sync_local=[OK]  bulk_sync_peer=[OK]
  dpu4:   keepalive=[OK]  bulk_sync_local=[OK]  bulk_sync_peer=[OK]
```

Membership failures

Membership failures can trigger a switchover/takeover scenario and can occur as a result DPU keep-alive failures, version and platform mismatches, load-balance mode mismatches.

In case both switches are FW ready, the leader takes over the traffic and the follower transitions to `not-ready`.

Traffic deflection mechanisms are enabled when the firewall transitions to `not-ready` while in `in-service`.

Step 1 The peer elected as leader takes over the traffic and the follower transitions local state as not-ready.

Example:

```
switch# show service-acceleration high-availability status
  Service System: hypershield
    HA Source Interface: loopback1 (192.1.1.1)
    HA Admin State: no-shutdown
    Agent Status for Service Firewall: not-ready     --> not ready
    Agent HA Status: switchover                      --> switched to peer
```

```
    Peers:
      PeerIP              Peer Service State  HA State with Peer  Reason
      ------------------- ------------------  ------------------  --------------
      192.1.1.2           ready               ha-fail             membership failure: peer_compatible
switch# show service-acceleration high-availability status
  Service System: hypershield
    HA Source Interface: loopback1 (192.1.1.2)
    HA Admin State: no-shutdown
    Agent Status for Service Firewall: ready     --> ready
    Agent HA Status: takeover                    --> takes over from peer
    Peers:
      PeerIP              Peer Service State  HA State with Peer  Reason
      ------------------- ------------------  ------------------  --------------
      192.1.1.1           not-ready           ha-fail             membership failure: peer_compatible
```

Step 2 Use this command to identify reasons for compatibility failures.

Example:

```
switch# service system hypershield connect session /usr/src/app/agwctl ha compatibility
Client got: ok
Data:
=== HA Compatibility ===
--- Peer: 192.1.1.1 ---
Membership:
Field          Local                          Peer                           Status
model          N9324C-SE1U                    N9324C-SE1U                    [OK]
nxos_version   10.6(3q)ISQ9(0.13)             10.6(3q)ISQ9(0.13)             [OK]
cpa_version    v1.19.0-pre.8-151-g14b22c12    v1.19.0-pre.8-151-g14b22c12    [OK]
dpu_count      4                              4                              [OK]
lb_mode        symmetric-hash                 pinning                        [FAIL]
policy_check   false                          false                          [OK]
dpu:dpu1       1.150.2                        1.150.2                        [OK]
dpu:dpu2       1.150.2                        1.150.2                        [OK]
dpu:dpu3       1.150.2                        1.150.2                        [OK]
dpu:dpu4       1.150.2                        1.150.2                        [OK]
Verdict: [FAIL]
Adjacency:
Field             Local   Peer   Status
policy_revision                  [OK]
Verdict: [OK]
```

Step 3 Verify that the VRFs are moved to redirection `installed` or `isolated` state and VRF isolation takes effect. The vPC traffic deflection also takes effect.

Example:

```
switch# show service-acceleration status details
  Service System: hypershield
    Source Interface: loopback2 (10.29.251.101)
    DPU Load-balance Mode: pinning
    Agent Status: firewall-ready,redirect-installed
    Agent Health Status: ok
    Controller Connection Status: init
    Agent HA Status: switchover
    Services:
      Firewall: in-service
      VRF                              Operational State       Affinity(DPU)
      ========================================================================
      tenant1                          redirection installed   1
      tenant2                          redirection installed   2
      VLAN                             Operational State       Affinity(DPU)
      ========================================================================
      2003                             forwarding ready        1
      2005                             forwarding ready        3
```

Step 4 Verify Hypershield Agent HA status during switchover or takeover.

Example:

```
switch# service system hypershield connect session /usr/src/app/agwctl ha show
Client got: ok
Data:
=== HA Status ===
Cluster State:   active/standby (degraded)
NX State:        ha-takeover
--- Local ---
  Svc State:         ready since 2026-05-12 20:15:48 UTC
    Reason:          all criteria met
    Criteria:        [OK] since 2026-05-12 20:15:48 UTC
      dpu_healthy:   [OK]
      dpu_insync:    [OK]
      in_service:    [OK]
--- Peer: 192.1.1.1 ---
  HA State:               ha-fail since 2026-05-13 00:30:34 UTC
    Reason:               membership failure: peer_compatible
  Svc State:              not-ready since 2026-05-13 00:30:17 UTC
    Reason:               peer service not-ready
    Criteria:             [FAIL] since 2026-05-13 00:30:17 UTC
      peer_service:       [FAIL]
  Membership:             [FAIL] since 2026-05-13 00:30:20 UTC
    peer_compatible:      [FAIL]
    peer_dpu_keepalive:   [OK]
    peer_vrf_gid:         [OK]
  Adjacency:              [FAIL] since 2026-05-13 00:30:24 UTC
    peer_dpu_bulk_sync:   [FAIL]
    peer_policy:          [OK]
```