ハイパーシールド付き Cisco N9300 スマートスイッチのトラブルシューティング

- DPU の起動 （1 ページ）
- HSC から HSA への接続の問題 （3 ページ）
- ファイアウォールの立ち上げ前のネットワーク構成 （7 ページ）
- ファイアウォールの立ち上げの NPU パケット フロー （8 ページ）
- ファイアウォールの起動 （10 ページ）
- パケット転送 （11 ページ）
- HSA と NXOS の相互作用 （13 ページ）
- NPU から DPU へのリダイレクトの問題 （14 ページ）
- DPU パケット フローを投稿 （16 ページ）
- パケット トレーサのサポート （17 ページ）

DPU の起動

スマート スイッチがセキュリティ機能を提供するには、DPU の正常な起動が必要です。これらのコマンドは、サービス アクセラレーション機能を有効にした後、DPU のステータスを確認します。

手順

ステップ 1 **service-acceleration** 機能が有効化されていることを確認します。

例：

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

ステップ 2 （任意） **show feature** コマンドを使用して、機能ステータスを確認します。

例：

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

ステップ 3 DPU の電源がオンでオンラインであることを確認します。

例：

```
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
```

リストされているすべての DPU **Status** が **ok**であることを確認します。

ステップ 4 DPU ソフトウェアとハードウェアのバージョンの確認します。

例：

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

各 DPU に記載されているソフトウェア（Sw）とハードウェア（Hw）のバージョンを確認します。

ステップ 5 現用系 DPU ファームウェア パッケージを確認します。

例：

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

ステップ 6 DME の DPU 初期化状態を確認します。

例：

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

initState が **inventory-done**であることを確認します。

ステップ 7 DPU からサービス イーサネット インターフェイスへのマッピング、および DPU IP アドレスを確認します。

例：

```
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
```

Number of DPUs と **Total number of DPU <-> NPU links**を確認します。リストされている各 DPU（**DPU-1**など）について、**Number of links to NPU** が 1 であること、 **Service Ethernet Interface(NPU)** が正しいインターフェイスにマッピングされること（たとえば、DPU-1 の場合は **SEth1/1**）、一意の **DPU-N IP address** が割り当てられていることを確認します（169.254.xx 内の範囲）。

これらのチェックを実行した後、DPUの電源が正常にオンになり、動作状態にあることを確認できます。

HSC から HSA への接続の問題

スイッチ上の Hypershield Agent が Hypershield Controller との接続を確立できない問題を診断し、解決します。HSC から HSA への接続をトラブルシューティングするための主要なツールは、一連のチェックを実行する **service system hypershield test controller connection** コマンドです。

手順

ステップ 1 コントローラ接続テスト電力事業を実行します。

例：

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

 ================================================
```

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

このコマンドは、いくつかのチェックを実行して、接続失敗のステータスと考えられる理由を表示します。

正常に接続されたら、次のことを確認します。

- **Agent Status** は、 **firewall-ready, redirect-installed** （完全なオンボーディング後）や同様の正常な状態など、準備ができていることを示します。
- **Controller Connection Status** は **[success]**です。
- ネットワーク接続チェック（DNS 解決、ping、curl）が成功しました。

例：

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

エージェントが実行されていない場合、そのことが出力に明示的に表示されます。

これを解決するには、 **service system hypershield** コマンドが構成されていることを確認します。

また、 **source-interface loopback <idx>** が **service system hypershield** の下で構成されており、構成されたループバック インターフェイスにデフォルトの VRF で有効な /32 IP アドレスがあることを確認します。

show system internal service-acceleration agent status を使用して、エージェントの状態を確認します。**started** と表示されている必要があります。

ステップ 2 ファイアウォール サブシステムが **in-service** でない場合は、出力を解釈します。

例：

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

ファイアウォール サービスが有効になっていない場合、**in-service** を構成するように出力に求められます。

これを解決するには、 **サービス システム ハイパーシールド サービス ファイアウォール** コンフィギュレーション モードで **in-service** を構成します。

ステップ 3 ワンタイム パスワード（OTP）が構成されていない場合の出力を解釈します。

例：

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

OTP が構成されていない場合、 出力にはトークンが欠落していることが示されます。

これを解決するには、Hypershield 管理ポータル（HS MP）から OTP を取得し、 **service system hypershield register <token>** EXEC コマンドを使用してスイッチ上で構成します。

sh system internal dme running-config all dn sys/sas/volatiledata/agent-hypershield | grep connToken を使用して、トークンが構成されているかどうかを確認できます（ただし、トークン自体ではありません）。

ステップ 4 HSA がハンドシェイクを完了していない場合に出力を解釈し、すべての DPU と同期します。

例：

```
switch(config-svc-sys-fw)# service system hypershield test controller connection
==============================
Running config checks
==============================
App Version: 1.1.0.rc.108
Health Status: failed
Hypershield Agent <-> DPU handshake pending. Collect 'show tech-support service-acceleration'
==================================
HSA DPU connection diagnostics
==================================
Client got: ok
Data:
IP 169.254.24.1, Number 2, InSync true, Healthy true
IP 169.254.28.1, Number 1, InSync false, Healthy true  -> DPU 1 InSync is false leading to failure

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

HSA が 1 つ以上の DPU と通信または同期できない場合、 **Health Status** は **failed**と表示され、出力は保留中のハンドシェイクを示すか、DPU 診断を提供します。

DPU 診断内：

- **InSync false** ファイアウォール ポリシーが DPU と HSA の間で同期していないことを示します。
- **Healthy false** DPU と HSA 間の定期的なハンドシェイクが失敗していることを示します。

このシナリオでさらにデバッグするには 、**show tech-support service-acceleration** を収集することを推奨します。

ステップ 5 必要なプロキシ構成が欠落しているか、ネットワーク接続の問題が存在する場合、出力を解釈します。

例：

```
switch(config-svc-sys-fw)# service system hypershield test controller connection
            ==============================
            Running config checks
            ==============================
            App Version: 1.1.0.rc.108
            Health Status: ok
            Using source-interface loopback1 for controller connection
            Controller connection token is configured
            Controller connection status: [init] -> HSA failed to connect to HSC
            Controller reason: [rpc error: code = Unavailable desc = connection error: desc =
"transport: Error while dialing: dial tcp 3.15.252.151:8880: i/o timeout"] -> Reason for the failureas
 reported by HSA
            ================================================
            Starting network connectivity checks on the switch
            ================================================
```

```
            Not using proxy for controller connection -> Indicates proxy not configured/used for
the connection between HSA and HSC
            Using curl to check connection to controller.prod.hypershield.engineering port 8880
           curl: (28) Failed to connect to controller.prod.hypershield.engineering port 8880 after
 8753 ms: Connection timed out
           Connection failed. Curl error: 28. Running DNS and ping test-> unable to connect to HSC
 using curl indicating a network  issue either because of missing configuration or incorrecttopology

            nslookup www.google.com to check DNS resolution.
            Server:    171.70.168.183
            Address 1: 171.70.168.183 dns-sj.cisco.com
            Name:      www.google.com
            Address 1: 142.250.189.228 nuq04s39-in-f4.1e100.net
            Address 2: 2607:f8b0:4005:80e::2004 nuq04s39-in-x04.1e100.net
            nslookup www.google.com using DNS 171.70.168.183 success. -> DNS validation successful
 indicating valid DNS and reachability to DNS
            Pinging controller.prod.hypershield.engineering to check network connectivity.
            PING controller.prod.hypershield.engineering (3.13.166.184) 56(84) bytes of data.
            --- controller.prod.hypershield.engineering ping statistics ---
            4 packets transmitted, 0 received, 100% packet loss, time 3033ms
            Could not ping controller.prod.hypershield.engineering. Checking for a valid route for
 controller.prod.hypershield.engineering ->  reachability to HSC using ping failed
            3.12.93.183 via 36.36.36.1 dev Eth1-5 src 36.36.36.2 uid 0
            cache
            controller.prod.hypershield.engineering has a valid route on the host. -> indicates a
valid route for HSC  on the switch
            Validating container networking on the switch.
            Container networking on switch successful. -> confirms successful container networking
 bring up
            controller.prod.hypershield.engineering has a valid route. Confirm if -> Failure could
 potentially be because of any of the 3 reasons
            1) controller.prod.hypershield.engineering is valid and can be reached without a proxy
            2) source-interface 10.29.251.4 is routable in the network
            3) ICMP is not blocked in network.
```

ネットワークの問題やプロキシ構成の不足が原因で HSA がコントローラに到達できない場合、**Controller connection status** が **[init]** であるか、「i/o timeout」などの RPC エラーが表示されている可能性があります。**curl** を使用したネットワーク接続チェックは、失敗します。

プロキシが必要な場合は、**service system hypershield** 構成モードの **https-proxy <hostname|IP> port <port_num>** を使用して構成されていることを確認します。

このテスト電力事業は、DNS テストおよび ping テストも実行します。出力を分析して、DNS 解決とコントローラのホスト名/ IPへの基本的な IP 到達可能性を確認します。

ステップ 6 （任意） ping を使用して HSA コンテナからの到達可能性をテストします。

例：

```
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
```

このコマンドは、pingがコンテナ内から実行されるため、HSAの観点から基本的なIP接続を確認します。ping を送信するコントローラの IP アドレスまたはホスト名を指定します。

service system hypershield test controller connection コマンドの出力を分析し、HSA コンテナから追加の ping テストを実行することで、HSA から HSC への接続を妨げている一般的な構成またはネットワークの問題を特定できます。

ファイアウォールの立ち上げ前のネットワーク構成

ファイアウォールサービスが構成されているものの、アクティブになっていない場合に、システムが想定どおりの状態であることを確認します。この状態では、サービスアクセラレーション用に構成された VRF は分離される必要があり、トラフィックは DPU にリダイレクトされないようにする必要があります。VRF が分離されているときにトラフィックが予期せず DPU に転送された場合、または VRF が分離されたときにボックス外の場合は、このセクションのチェックを実行します。

手順

ステップ 1 **service firewall** が作成され、 **out-of-service** 状態であることを確認します。

例：

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

これにより、ファイアウォール サービスが構成されていることを確認します。

ステップ 2 ファイアウォール サービスの動作ステータスを確認します。

例：

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

Firewall 状態は **out-of-service**になります。

ステップ 3 サービス アクセラレーション用に有効になっている VRF を特定します。

例：

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

service firewall セクションで **vrf <name>**で始まる行をチェックして、サービス アクセラレーション用に構成された VRF を特定します。

ステップ 4 識別された VRF が **isolated** 状態であることを確認します。

例：

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

構成された各 VRF の **Operational State** が **isolated**であることを確認します。

VRFのルーティング プロトコルによってアドバタイズされたルートが、プロトコル分離中にサポートされている動作と一致することを確認します。

（注）
VRF 分離動作は、現在 OSPF と BGP でサポートされています。

これらのチェックを実行した後、サービス アクセラレーション用に構成された VRF がシステムおよびルーティングプロトコルによって分離されていると正しく識別されることを確認できるはずです。これは、ファイアウォールサービスが起動（インサービス）になる前の予想される状態です。

ファイアウォールの立ち上げの NPU パケット フロー

ファイアウォールサービスがアクティブでない（サービス停止状態）場合、サービスアクセラレーション VRF 宛てのトラフィックは、（BFDやマルチキャストなどの特定のプロトコルについては）ドロップまたはバイパスする必要があり、DPUにリダイレクトしないでください。次の手順で、この動作を確認します。VRF の状態が正しく、クリーンな整合性チェックを行っているにもかかわらず、トラフィックが予期せず DPU に転送されるか、またはボックス外の場合は、このセクションの確認手順を実行します。

手順

ステップ 1 特定の VRF の整合性チェッカを実行して、トラフィック処理の潜在的な問題を特定します。

例：

```
switch# show consistency-checker service-acceleration vrf red
            VRF 'red' is configured in Service-acceleration
```

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

red をVRF名に置き換えます。このチェックでは、構成と動作状態に一貫性があるかどうか、および必要なポリシー適用ポイントが配置されているかどうかを確認します。**out-of-service** 状態で、動作状態が **isolated** であることを確認します。

ステップ 2 VRFのインターフェイスに対して生成された内部ポリシーがBFDエコーおよびマルチキャストトラフィックを許可し、他のすべての IP トラフィックをドロップすることを確認します。

例：

```
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
```

この例は、内部で生成された IPv4 ポリシーを示しています。ACL カウンタを確認して、アクセスリストと一致するパケットがあるかどうかを検出します。

ステップ 3 （任意） サービス イーサネット インターフェイスのカウンタをチェックして、トラフィックが DPU に送信されていないことを確認します。

例：

```
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
```

out-of-service 状態の **DPU** にトラフィックを送信してはならないため、これらのインターフェイスのカウンタがゼロのままであるか、または最小限の制御トラフィックだけを表示することを確認します。

これらのチェックを実行することにより、NPUがアウトオブサービス状態のサービスアクセラレーション VRF のトラフィックを正しく処理していること、プログラムされた Pass-1 ACL に従ってトラフィックがドロップまたはバイパスしていること、およびトラフィックが DPU に送信されていないことを確認できます。

ファイアウォールの起動

ファイアウォール サービスと構成済みの VRF が、有効になった後に、予想される動作状態に遷移したことを確認します。サービス ファイアウォールは、 **in-service** コマンドの実行後、非同期的に準備完了になります。次の手順により、システムがファイアウォールのためにトラフィックを引き付けて処理する準備が整っていることが確認されます。

手順

ステップ 1 **in-service** コマンドがファイアウォール サービス用に構成されていることを確認します。

例：

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

in-service 行が **サービス ファイアウォール** 構成の下にあることを確認します。

ステップ 2 HSA と構成された VRF の最終状態を確認します。

例：

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

このコマンドは詳細なステータスを提供します。次のことを確認してください。

- **Agent Status** は、準備ができていることを示します（**firewall-ready,redirect-installed**など）。
- **Controller Connection Status** は **success**です。
- **Firewall** サービスのステータスは **in-service** です。
- 構成された各VRFには **forwarding ready**、の **Operational State** が表示されます。

• 各VRFの **Affinity(DPU)** は、構成されたアフィニティと一致する有効な DPU 番号を示しています。

これらのチェックを実行すると、ファイアウォール サービスがアクティブであり、HSA と構成済みの VRF を含むシステムが、ファイアウォールのトラフィックの処理を開始するために必要な動作状態に達していることを確認できます。

パケット転送

show トラブルシュート l3 コマンドを使用して、サービス アクセラレーション リダイレクションを含む、特定の IP フローの転送パスを確認します。このコマンドは、さまざまなソフトウェアおよびハードウェア テーブルのルーティング情報をチェックし、リダイレクションの問題のトラブルシュートに役立つサービス アクセラレーション整合性チェックを含めます。

手順

ステップ 1 特定の IPv4 フローに対して L3 トラブルシューティング コマンドを実行します。

例：

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

```
SWITCH TYPE: TOR

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

トラブルシューティングするフローの接続先IP（**181.1.1.3**）、送信元IP（**180.1.1.3**）、およびVRF名（**red**）を指定します。

次の項目が含まれる出力を分析します：

- 指定された VRF のサービス アクセラレーション整合性チェックします。
- プロトコル独立 RIB（PI RIB）、プロトコル依存 FIB （PD FIB）、および統合 FIB （UFIB）でのルートルック アップ。
- ハードウェアのホスト ルートとネクストホップをチェックします。
- 最終的な整合性チェッカーの実行。

整合性チェックで **SUCCESS** メッセージが表示され、ルート ルックアップで予想されるネクストホップまたはリダイレクション パスが特定されていることを確認します。

ステップ 2 （任意） 特定の IPv6 フローに対して L3 トラブルシューティング コマンドを実行します。

例：

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

このコマンドを活用、IPv6 パケット転送の問題をトラブルシュート、必要に応じて IP アドレスと VRF 名を置き換えます。

---

how troubleshoot l3 コマンドが正常に完了し、ルートについて「Consistency checker passed」というメッセージが表示される場合は、DPUへのリダイレクションを含め、指定されたパケット

フローを転送するために必要なルーティングおよびサービス アクセラレーション構成がシステムにあることを示しています。

HSA と NXOS の相互作用

Hypershield Agent（HSA）が、EPBR を使用したトラフィック リダイレクションに必要な構成オブジェクトをデータ管理エンジン（DME）を介して NX-OS に正常にプッシュしたことを確認します。クリーンな整合性チェックにもかかわらず、トラフィックがスイッチに到着し、DPU をドロップまたはバイパスする場合は、このセクションの手順を使用してスイッチのリダイレクション情報を確認します。

手順

エージェントがリダイレクト ポリシーとサービス構成を NX-OS に送信したかどうかを確認します。

例：

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

Hypershield Agent によってプッシュされたダイナミック EPBR 構成を確認します。EPBR ポリシーが正しい VRF に関連付けられていること、および EPBR サービスが各 VRF の正しい DPU モジュールとカプセル化 VLAN ID を定義していることを確認します。

これらのチェックを実行することにより、Hypershield Agent が必要な EPBR 構成オブジェクトを DME を介して NX-OS に正常にプッシュしていることを確認し、サービス アクセラレーションされた VRF のトラフィックを適切な DPU にリダイレクトする方法を定義できます。

NPU から DPU へのリダイレクトの問題

EPBR リダイレクション ポリシーが NPU ハードウェア（TCAM）で正しくプログラムされていること、およびトラフィックがサービスイーサネットサブインターフェイスを介して適切な DPU に正常にリダイレクトされていることを確認します。これらのチェックは、ポリシーがエージェントによって正しくプッシュされている場合でも、トラフィックが着信しているが DPU にリダイレクトされていないか、または誤った DPU に送信されている場合に、原因を特定するのに役立ちます。

手順

ステップ 1 VRF ごとに作成されたサービス サブインターフェイスを識別して、 VRFトラフィックを DPU にリダイレクトし、それらのステータスを確認します。

例：

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

VRF およびアドレス ファミリごとに作成されたサービス サブインターフェイス（**Interface[Status]**）を識別します。インターフェイスのステータスが **UP** であることおよび **Redirect Status** が **Enabled**であることを確認します。

ステップ 2 特定の service-ethernet サブインターフェイスの構成の詳細を確認します。

例：

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

前のステップで特定したサブインターフェイス（**1/1.11**）を指定します。**VLAN** ID がVRFの EPBR サービスで構成されたものと一致すること、 **Mode** が **routed**、 **Status** が **up**であることを確認します。

ステップ 3 特定の VRF の EPBR で詳細なポリシー プログラミング情報を確認します。

例：

```
switch# show service-acceleration redirect-policy vrf red
 VRF: red
 Policy-map: __red_dpu_redir
```

```
Default Traffic Action: Redirect
Match clause:
ip address(access-lists): __dpu_ipv6_redir
action: Redirect
status(inside): CREATED/ENABLED         status(post_fwd): CREATED/ENABLED
service __red_dpu_redir, sequence 10, fail-action Drop
service-module 1 vlan 11 [UP] [SEth1/1.11]
Match clause:
ip address(access-lists): __dpu_redir
action: Redirect
status(inside): CREATED/ENABLED         status(post_fwd): CREATED/ENABLED
service __red_dpu_redir, sequence 10, fail-action Drop
service-module 1 vlan 11 [UP] [SEth1/1.11]
```

red をVRF名に置き換えます。match 句の **status(inside)** と **status(post_fwd)** に **CREATED/ENABLED**が表示されていることを確認します。**service-module**、**vlan**、およびサービス インターフェイス（**SEth1/1.11**）の情報が正しく、この VRF で想定される DPU およびサブインターフェイスと一致していることを確認します。

ステップ 4 VRF の内部で生成されたポリシーと一致するパケット カウンタを確認します。

例：

（IPv4 トラフィック）

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

例：

（IPv6 トラフィック）

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

上記のルールで VNI として使用される **show vrf <> detail** を介して VRF のテーブル ID を特定して、対象の VRF に関連するルールを特定します。ルールに、VRF の適切なサブインターフェイスへのリダイレクションが表示されていることを確認します。VRF ルールのパケット ヒット カウンタを確認します。

ステップ 5 （任意） サービス アクセラレーション リダイレクト ポリシーのリソース使用率を確認します。

例：

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

```
  LBL AP    Ingress IPv6 SVC Redir L3 Policy Pass-1          2       125     1.57
  LBL AQ    Ingress IPv4 SVC Redir L3 Policy Pass-2          2       125     1.57
  LBL AR    Ingress IPv6 SVC Redir L3 Policy Pass-2          2       125     1.57
```

Pass-1 および **Pass-2** リダイレクト ポリシーを含む ACL ポリシーのハードウェアウェア リソースの使用状況を確認します。高い使用率またはエラーは、リソースの枯渇を示している可能性があります。

これらのチェックを実行することにより、サービスが高速化されたVRFのトラフィックをサービス イーサネット サブインターフェイスを介して適切なDPUにリダイレクトするようにNPUが正しくプログラムされていることを確認し、トラフィックが予想されるハードウェア ルールにヒットしていることを確認できます。

DPU パケット フローを投稿

サービス イーサネット インターフェイスに適用される Pass-2 ACL をチェックし、インターフェイス構成の詳細を確認して、DPU から戻るトラフィックが接続先への転送のために NPU によって正しく処理されていることを確認します。トラフィックが NPU でドロップしているか、DPUから戻った後にルーティングされない場合は、このセクションの確認手順を実行します。

手順

ステップ 1 フローを学習する前に、検査の2番目のパスでDPUにリダイレクトするVRFの内部で生成されたポリシーと一致するパケット カウンタを確認します。

例：

（IPv4 トラフィック）

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

例：

（IPv6 トラフィック）

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

上記のルールで VNI として使用される **show vrf < > detail**を介して VRF のテーブル ID を特定して、対象の VRF に関連するルールを特定します。ルールに DPU から返されるトラフィックに一致する permit エント

リが含まれていることを確認し、そのトラフィックをVRFの正しいサービスサブインターフェイスにリダイレクトします。これらの VRF ルールのパケット ヒット カウンタを確認します。

ステップ 2 DPU 接続サービス イーサネット ポートの統計を表示します。

例：

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

これらのコマンドは、パケット/バイト カウンタ、詳細統計情報、エラー統計情報、平均レートなど、DPU に接続されているサービス イーサネット インターフェイスのさまざまな統計情報を提供します。これらのカウンタをチェックすると、トラフィックが予想どおりにDPUに到達しているかどうかを判断できます。

これらのチェックを実行することにより、NPU が DPU からの戻りトラフィックを処理するように Pass-2 ACL を正しくプログラムしていること、およびサービス イーサネット インターフェイスとサブインターフェイスが転送用の正しいパラメータで構成されていることを確認できます。

パケット トレーサのサポート

パケット トレーサ電力事業を活用、エンドツーエンドの NPU から DPU への NPU パケット フローのさまざまなポイントでパケットをキャプチャして分析します。これにより、さまざまな段階でトラフィック処理の詳細なトラブルシューティングが可能になります。

次の手順は、特定のポイントでパケットをキャプチャする方法の概要を示しています。

- NPU 入力
- NPU から DPU TX へ
- DPU から NPU RX へ：最初のパケット
- DPU から NPU RX へ：後続のパケット
- NPU の出力

手順

ステップ 1 入力 L3 インターフェイス（NPU 入力）で受信されたパケットをキャプチャします。

例：

```
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
```

これは、DPUにリダイレクトする前に、入力レイヤ3インターフェイスでNPUが受信したパケットをキャプチャします。

ステップ2 サービス イーサネット インターフェイス（SETH TX）上の NPU から DPU に送信されたパケットをキャプチャします。

例：

```
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
```

これは、NPU からサービス イーサネット インターフェイス上の DPU に送信されたパケットをキャプチャします。

（注）
スナップショットは DPU ヘッダーのカプセル化が行われる前に取得されるため、この時点では DPU ヘッダーの詳細を追跡またはキャプチャすることはできません。

ステップ3 パスビットが設定された（SETH RX dpu pass 0）状態で NPU が DPU から受信した最初のパケットをキャプチャします。

例：

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

これは、サービスイーサネット インターフェイスの DPU から NPU が受信した最初のパケット、特に DPU パスビット 0 でマークされたパケット（通常、DPU のコントロール プレーンにパントされるフローの最初のパケット）をキャプチャします。

（注）
スナップショットは DCPU のカプセル化が解除される前に取得されますが、受信した DPU ヘッダーのインサイトを得ることできます。

ステップ4 パスビットが設定された状態（SETH RX dpu pass 1）で DPU から NPU が受信した後続のパケットをキャプチャします。

例：

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

これは、サービスイーサネット インターフェイスの DPU から NPU が受信したフローの後続パケットをキャプチャします。通常、それらは DPU のデータ プレーン（P4）によって直接処理され、DPU パスビット 1 でマークされます。

（注）
スナップショットは DCPU のカプセル化が解除される前に取得されますが、受信した DPU ヘッダーのインサイトを得ることできます。

ステップ 5 （任意） IP アドレスに基づいて DPU から NPU が受信した特定のパケットをキャプチャします（SETH RX dpu パス 1 - 特定のフロー）。

例：

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

これにより、NPU が DPU から受信した特定のフローの後続パケットがキャプチャされます。これは、DPU フィルタと外部 IP アドレス フィルタを組み合わせたものです。

ステップ 6 ルーティング（NPU 出力）後に NPU から出力 L3 インターフェイスに送信されたパケットをキャプチャします。

例：

```
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
```

これは、DPU 後に処理およびルーティングされた後に、出力レイヤ 3 インターフェイスに向けて NPU によって送信されたパケットをキャプチャします。

（注）

スナップショットは DPU ヘッダーのカプセル化が行われる前に取得されるため、この時点では DPU ヘッダーの詳細を追跡またはキャプチャできないことに注意してください。

これらのパケットキャプチャ手順を実行した後、キャプチャされたパケットを分析して、NPU から DPU への NPU フローのさまざまなポイントでトラフィックがどのように処理されているかのインサイトを得ることができます。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。