ACI ファブリックのコントラクトおよびルールの確認

内容

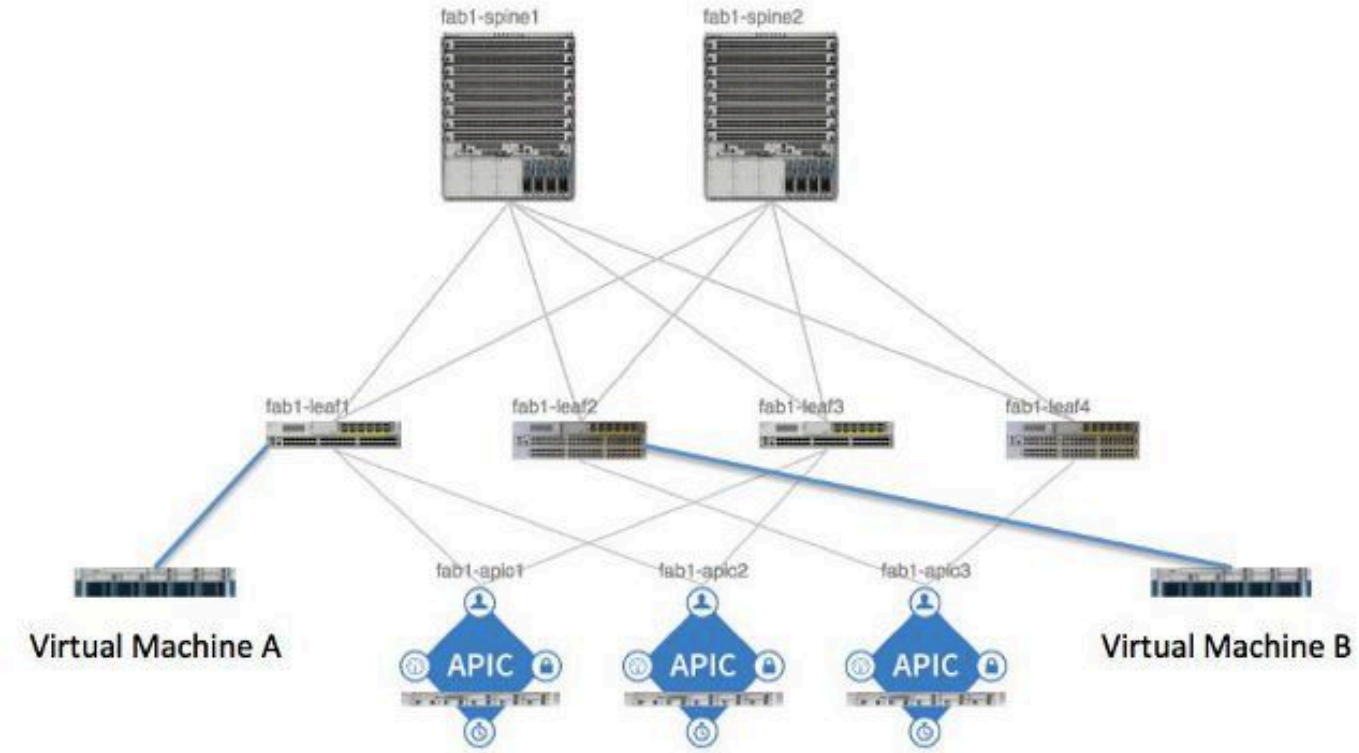
はじめに

このドキュメントでは、アプリケーション セントリック インフラストラクチャ（ACI）ファブリックにコントラクトが設定されていて、正常に動作していることを検証する方法を説明します。

トポロジ

このドキュメント全体を通して使用する例では、仮想マシン（VM）A がリーフ 1 に接続され、リーフ 2 に接続された VM-B との通信を許可するコントラクトが設定されています。このコントラクトでは、Internet Control Message Protocol（ICMP）および HTTP の両方が許可されます。

以下の図に、トポロジを示します。



プロセスの概要

コントラクトとルールのポリシー インタラクションおよびフローは以下のとおりです。

1. Application Policy Infrastructure Controller（APIC）上のポリシー マネージャは、スイッチ上のポリシー要素マネージャと通信します。

2. スイッチ上のポリシー要素マネージャは、スイッチ上のオブジェクト ストアをプログラムします。

3. スイッチ上のポリシー マネージャは、スイッチ上のアクセス コントロール リスト QoS（ACLQOS）クライアントと通信します。

4. ACLQOS クライアントは、ハードウェアをプログラムします。

使用されたコントラクト/ゾーニング ルールの特定

以下に、2つのエンドポイントグループ(EPG)に対するコントラクトが追加される前にリーフに対して実行したshow zoning-ruleコマンドの出力例を記載します。

<#root>

```
fab1_leaf1#
```

show zoning-rule

```
Rule ID  SrcEPG  DstEPG  FilterID  operSt   Scope     Action
=======  ======  ======  ========  ======   =====     ======
4096     0       0       implicit  enabled  16777200  deny,log
4097     0       0       implicit  enabled  3080192   deny,log
4098     0       0       implicit  enabled  2686976   deny,log
4099     0       49154   implicit  enabled  2686976   permit
4102     0       0       implicit  enabled  2097152   deny,log
4103     0       32771   implicit  enabled  2097152   permit
4117     16387   16386   12        enabled  2097152   permit
4116     16386   16387   13        enabled  2097152   permit
4100     16386   49154   default   enabled  2097152   permit
4101     49154   16386   default   enabled  2097152   permit
4104     0       32770   implicit  enabled  2097152   permit
4105     49155   16387   13        enabled  2097152   permit
4112     16387   49155   13        enabled  2097152   permit
4113     49155   16387   12        enabled  2097152   permit
4114     16387   49155   12        enabled  2097152   permit
[snip]
```

コントラクトが追加され、2 つの EPG が相互通信できるようになった後に同じコマンドを実行すると、以下の出力が生成されます。

```
<#root>
fab1_leaf1#
show zoning-rule
```

```
Rule ID  SrcEPG  DstEPG  FilterID  operSt   Scope     Action
=======  ======  ======  ========  ======   ========  ========
4096     0       0       implicit  enabled  16777200  deny,log
4097     0       0       implicit  enabled  3080192   deny,log
4098     0       0       implicit  enabled  2686976   deny,log
4099     0       49154   implicit  enabled  2686976   permit
```

```
4131     49155   32771   7         enabled  2686976   permit


4132     32771   49155   6         enabled  2686976   permit


4102     0       0       implicit  enabled  2097152   deny,log
4103     0       32771   implicit  enabled  2097152   permit
4117     16387   16386   12        enabled  2097152   permit
4116     16386   16387   13        enabled  2097152   permit
4100     16386   49154   default   enabled  2097152   permit
4101     49154   16386   default   enabled  2097152   permit
4104     0       32770   implicit  enabled  2097152   permit
4105     49155   16387   13        enabled  2097152   permit
4112     16387   49155   13        enabled  2097152   permit
4113     49155   16387   12        enabled  2097152   permit
4114     16387   49155   12        enabled  2097152   permit
[snip]
```


注：追加された新しいルールID（4131および4132）、フィルタIDが7および6、範囲が2686976であることに注意してください。

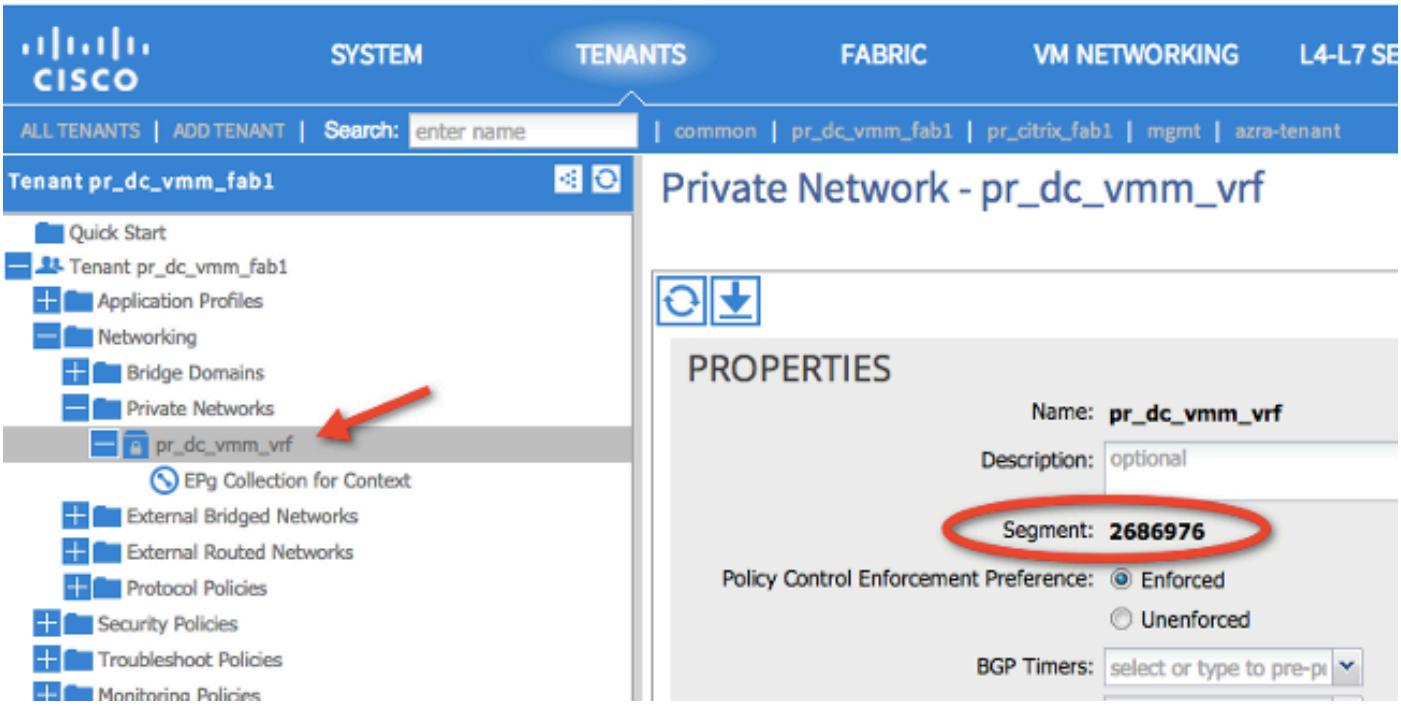
注意：このコマンド出力では、ラボシステムで確認する必要があるルールを簡単に見つけることができます。ただし、動的に変更が加えられるため、実稼働環境では確認が難しくなる可能性があります。

対象のルールを特定するために使用できる別の方法は、Visore を使用することです。コンテキスト管理対象オブジェクト（MO）で fvCtx を検索します。検索結果の画面で、特定のコンテキスト識別名（DN）を検索できます（以下を参照）。

そのコンテキストのスコープを書き留めます。このスコープを使用して show-zoning-rule コマンド出力をマッピングすることで、照会すべきルールを特定できます。

fvCtx	
childAction	
descr	
dn	uni/tn-pr_dc_vmm_fab1/ctx-pr_dc_vmm_vrf
knwMcastAct	permit
lcOwn	local
modTs	2014-09-03T09:32:36.625-04:00
monPolDn	uni/tn-common/monepg-default
name	pr_dc_vmm_vrf
ownerKey	
ownerTag	
pcEnfPref	enforced
pcTag	32770
scope	2686976
seg	2686976
status	
uid	15374

また、ユーザ インターフェイス（UI）でコンテキストのセグメント ID/スコープを特定することもできます（以下を参照）。



このスコープは、show zoning-rules コマンド出力に示されるスコープと一致します。

スコープ ID 情報を入手し、ルールとフィルタ ID を特定した後は、次のコマンドを使用して、新しいフィルタと一致すること（および出力に EPG 間の暗黙の deny メッセージが含まれていないこと）を検証できます。暗黙の deny メッセージが含まれている場合、デフォルトでは EPG が相互通信できません。

以下のコマンド出力で、リーフ 1 のフィルタ 6（f-6）が増加していることに注目してください。

<#root>

```
fab1_leaf1#
```

show system internal policy-mgr stats | grep 2686976

```
Rule (4098) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-any-f-implicit)
 Ingress: 0, Egress: 81553

Rule (4099) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-49154-f-implicit)
 Ingress: 0, Egress: 0

Rule (4131) DN (sys/actrl/scope-2686976/rule-2686976-s-49155-d-32771-f-7)
 Ingress: 0, Egress: 0
```

Rule (4132) DN (sys/actrl/scope-2686976/rule-2686976-s-32771-d-49155-f-6)
Ingress: 1440, Egress: 0

<#root>

fab1_leaf1#

show system internal policy-mgr stats | grep 2686976

Rule (4098) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-any-f-implicit)
 Ingress: 0, Egress: 81553

Rule (4099) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-49154-f-implicit)
 Ingress: 0, Egress: 0

Rule (4131) DN (sys/actrl/scope-2686976/rule-2686976-s-49155-d-32771-f-7)
 Ingress: 0, Egress: 0

Rule (4132) DN (sys/actrl/scope-2686976/rule-2686976-s-32771-d-49155-f-6)

Ingress: 1470, Egress: 0

以下のコマンド出力で、リーフ 2 のフィルタ 7 (f-7) が増加していることに注目してください。

<#root>

fab1_leaf2#

show system internal policy-mgr stats | grep 268697

Rule (4098) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-any-f-implicit)
 Ingress: 0, Egress: 80257

Rule (4099) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-49153-f-implicit)
 Ingress: 0, Egress: 0

Rule (4117) DN (sys/actrl/scope-2686976/rule-2686976-s-32771-d-49155-f-6)
 Ingress: 0, Egress: 0

Rule (4118) DN (sys/actrl/scope-2686976/rule-2686976-s-49155-d-32771-f-7)
Ingress: 2481, Egress: 0

<#root>

fab1_leaf2#

show system internal policy-mgr stats | grep 268697

Rule (4098) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-any-f-implicit)
 Ingress: 0, Egress: 80257

```
Rule (4099) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-49153-f-implicit)
 Ingress: 0, Egress: 0

Rule (4117) DN (sys/actrl/scope-2686976/rule-2686976-s-32771-d-49155-f-6)
 Ingress: 0, Egress: 0


Rule (4118) DN (sys/actrl/scope-2686976/rule-2686976-s-49155-d-32771-f-7)
 Ingress: 2511, Egress: 0
```

ヒント：この問題をさらにトラブルシューティングするには、スコープ、ルールID、宛先、送信元のpcTag、およびフィルタに関する知識が重要です。また、その間にルール ID が存在する EPG の知識も役立ちます。

DN 名が fvAEPg の MO に対して検索を実行し、moquery コマンドを使用して特定の pcTag を grep できます (以下を参照) 。

<#root>

```
admin@RTP_Apic1:~>

moquery -c fvAEPg | grep 49155 -B 5


dn : uni/tn-Prod/ap-commerceworkspace/

epg-Web


lcOwn : local
matchT : AtleastOne
modTs : 2014-10-16T01:27:35.355-04:00
monPolDn : uni/tn-common/monepg-default

pcTag : 49155
```

また、次に示すように、filter オプションを指定した moquery コマンドを使用することもできます。

<#root>

```
admin@RTP_Apic1:~>

moquery -c fvAEPg -f 'fv.AEPg.pcTag=="49155"'


Total Objects shown: 1

# fv.AEPg
name : Web
childAction :
configIssues :
configSt : applied
```

```
descr :
dn : uni/tn-Prod/ap-commerceworkspace/
```

epg-Web

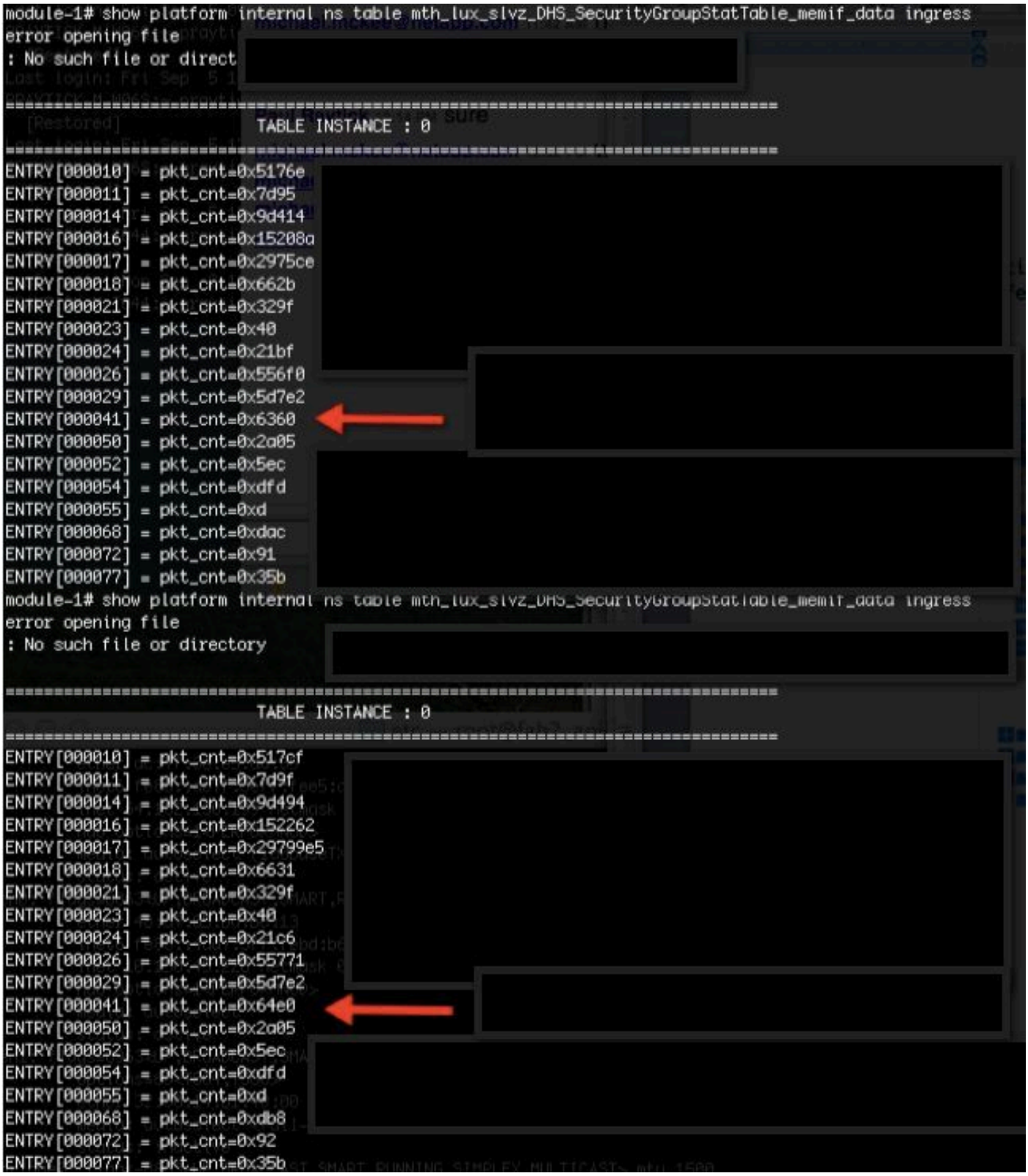
```
lcOwn : local
matchT : AtleastOne
modTs : 2014-10-16T01:27:35.355-04:00
monPolDn : uni/tn-common/monepg-default
```

pcTag : 49155

```
prio : unspecified
rn : epg-Web
scope : 2523136
status :
triggerSt : triggerable
uid : 15374
```

ハードウェア プログラミングの検証

次に、ルールのハードウェア エントリを確認します。ハードウェア情報を表示するには、show platform internal ns table mth_lux_slvz_DHS_SecurityGroupStatTable_memif_data ingress コマンドを入力します（これは vsh_lc コマンドです）。



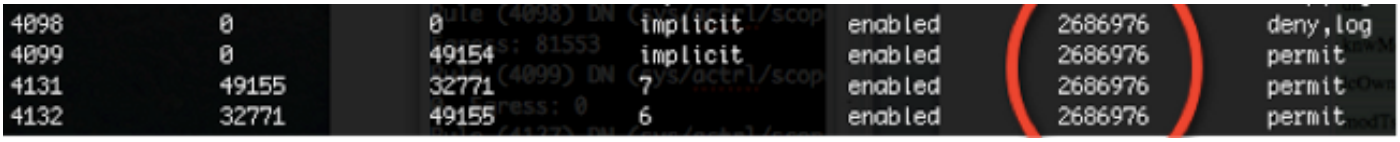
上記の例では、ハードウェア エントリ 41（ENTRY [000041]）が増加しています。

注：上記のコマンドはNorthstar ASIC用です。DonnerまたはDonner+に使用されるコマンドは、show platform internal ns table mth_luxh_slvy_DHS_SecurityGroupStatTable_memif_dataです。

注：このコマンドは、実稼働環境では実用的ではありませんが、このセクションで説明する

他のコマンドを代わりに使用できます。

ルール(4132)とスコープ(268976)を覚えておいてください。



ルール ID と Ternary Content-Addressable Memory（TCAM）ハードウェア インデックス エントリのマッピングを判別し、ルール ID/フィルタ ID を基準にフィルタリングするために、以下のコマンドを入力します。

<#root>

module-1#

show system internal aclqos zoning-rules

[snip]

```
==========================================
Rule ID: 4131 Scope 4 Src EPG: 49155 Dst EPG: 32771 Filter 7

Curr TCAM resource:
============================
   unit_id: 0
   === Region priority: 771 (rule prio: 3 entry: 3)===
       sw_index = 62 |
```

hw_index = 40

```
   === Region priority: 772 (rule prio: 3 entry: 4)===
       sw_index = 63 |
```

hw_index = 45

```
==========================================
Rule ID: 4132 Scope 4 Src EPG: 32771 Dst EPG: 49155 Filter 6

Curr TCAM resource:
============================
   unit_id: 0
   === Region priority: 771 (rule prio: 3 entry: 3)===
       sw_index = 66 |
```

hw_index = 41

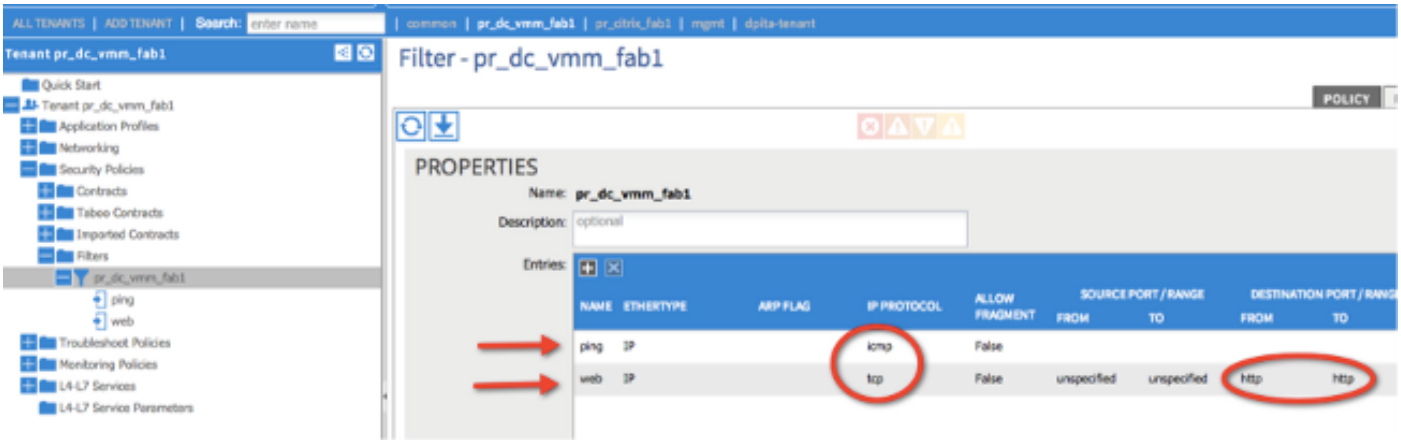
```
   === Region priority: 771 (rule prio: 3 entry: 3)===
       sw_index = 67 |
```

hw_index = 42

```
[snip]
```

この例では、対象の送信元と宛先のEPGの組み合わせは32771=0x8003、49155=0xC003です。したがって、ルールID（4131および4132）とフィルタID（6および7）に一致する、これらの送信元クラスと宛先クラスのすべてのTCAMエントリを考慮できます。

以下の例では、これらの TCAM エントリの一部がダンプされています。以下に参考として、これらの EPG に対して ping と Web トラフィックを許可するコントラクト設定を示します。



<#root>

```
module-1#
```

show platform internal ns table mth_lux_slvz_DHS_SecurityGroupKeyTable0

_memif_data 41

```
========================================================================
                        TABLE INSTANCE : 0
========================================================================
ENTRY[000041] =
               sg_label=0x4

               sclass=0x8003

               dclass=0xc003

               prot=0x1
```

(IP Protocol 0x01 = ICMP)

注：上記のコマンドはNorthstar ASIC用です。Donner または Donner+ に使用されるコマンドは、show platform internal ns table

mth_luxh_slvq_DHS_SecurityGroupKeyTable0_memif_data です。

Decimal	Keyword	Protocol	IPv6 Extension Header	
0	HOPOPT	IPv6 Hop-by-Hop Option	Y	[RFC2460]
1	ICMP	Internet Control Message		[RFC792]
2	IGMP	Internet Group Management		[RFC1112]

<#root>

```
sup_tx_mask=0x1
                src_policy_incomplete_mask=0x1

                dst_policy_incomplete_mask=0x1

                class_eq_mask=0x1

                aclass_mask=0x1ff

                port_dir_mask=0x1

                dport_mask=0xffff

                sport_mask=0xffff

                tcpflags_mask=0xff

                ip_opt_mask=0x1

                ipv6_route_mask=0x1

                ip_fragment_mask=0x1

                ip_frag_offset0_mask=0x1

                ip_frag_offset1_mask=0x1

                ip_mf_mask=0x1

                l4_partial_mask=0x1

                dst_local_mask=0x1

                routeable_mask=0x1

                spare_mask=0x7ff

                v4addr_key_mask=0x1

                v6addr_key_mask=0x1

                valid=0x1

module-1#
```

show platform internal ns table mth_lux_slvz_DHS_SecurityGroupKeyTable0

_memif_data 42

```
==========================================================================
                          TABLE INSTANCE : 0
==========================================================================
ENTRY[000042] =

                sg_label=0x4

                sclass=0x8003

                dclass=0xc003

                prot=0x6
<--


                dport=0x50
<--
```

Decimal	Keyword	Protocol	IPv6 Extension Header	
0	HOPOPT	IPv6 Hop-by-Hop Option	Y	[RFC2460]
1	ICMP	Internet Control Message		[RFC792]
2	IGMP	Internet Group Management		[RFC1112]
3	GGP	Gateway-to-Gateway		[RFC823]
4	IPv4	IPv4 encapsulation		[RFC2003]
5	ST	Stream		[RFC1190][RFC1819]
6	TCP	Transmission Control		[RFC793]
7	CBT	CBT		[Tony_Ballardie]

Port	TCP	UDP	Description
0	TCP		Programming technique for specifying system-allocated (dynamic) ports[3]
21	TCP		FTP control (command)
25	TCP		Simple Mail Transfer Protocol (SMTP)—used for e-mail routing between mail servers
43	TCP		WHOIS protocol
57	TCP		Mail Transfer Protocol (RFC 780)
70	TCP		Gopher protocol
71	TCP		NETRJS protocol
72	TCP		NETRJS protocol
73	TCP		NETRJS protocol
74	TCP		NETRJS protocol
79	TCP		Finger protocol
80	TCP		Hypertext Transfer Protocol (HTTP)[12]
81	TCP		Tornark – Onion routing

```
sup_tx_mask=0x1
                src_policy_incomplete_mask=0x1
                dst_policy_incomplete_mask=0x1
                class_eq_mask=0x1
                aclass_mask=0x1ff
                port_dir_mask=0x1
                sport_mask=0xffff
                tcpflags_mask=0xff
                ip_opt_mask=0x1
                ipv6_route_mask=0x1
                ip_fragment_mask=0x1
                ip_frag_offset0_mask=0x1
                ip_frag_offset1_mask=0x1
                ip_mf_mask=0x1
                l4_partial_mask=0x1
                dst_local_mask=0x1
```

ヒント：同じ方法で各TCAMエントリを確認できます。

ハードウェア プログラミングに関する問題のトラブルシューティング

ここでは、トラブルシューティングに役立つコマンドとヒントについて説明します。

有効なトラブルシューティング コマンド

問題が発生した際にリーフのポリシー マネージャ エラーを特定するには、以下のコマンドが役立ちます。

<#root>

```
fab1_leaf1#

show system internal policy-mgr event-history errors



1) Event:E_DEBUG, length:84, at 6132 usecs after Mon Sep 8 13:15:56 2014

   [103] policy_mgr_handle_ctx_mrules(779): ERROR: Failed to process prio(1537):
   (null)


2) Event:E_DEBUG, length:141, at 6105 usecs after Mon Sep 8 13:15:56 2014

   [103] policy_mgr_process_mrule_prio_aces(646): ERROR: Failed to insert iptables
   rule for rule(4120) , fentry(5_0) with priority(1537): (null)


[snip]


fab1_leaf1#

show system internal policy-mgr event-histor trace



[1409945922.23737] policy_mgr_ppf_hdl_close_state:562: Got close state callback

[1409945922.23696] policy_mgr_ppf_rdy_ntf_fun:239: StatStoreEnd returned: 0x0(SU

CCESS)

[1409945922.23502] policy_mgr_ppf_rdy_ntf_fun:208: ppf ready notification: sess_

id: (0xFF0104B400005B51)

[1409945922.23475] policy_mgr_ppf_rdy_ntf_fun:205: Got ready notification callba

ck with statustype (4)
```

```
[1409945921.983476] policy_mgr_gwrap_handler:992: Dropped...now purging it...

[1409945921.982882] policy_mgr_ppf_goto_state_fun:481: Sess id (0xFF0104B400005B

[snip]

module-1#
show system internal aclqos event-history trace


T [Fri Sep 5 13:18:24.863283] ============= Session End ============

T [Fri Sep 5 13:18:24.862924] Commit phase: Time taken 0.62 ms, usr 0.00 ms,

sys 0.00 ms

T [Fri Sep 5 13:18:24.862302] ppf session [0xff0104b410000087] commit ... npi

nst 1

T [Fri Sep 5 13:18:24.861421] Verify phase: Time taken 0.77 ms, usr 0.00 ms,

sys 0.00 ms

T [Fri Sep 5 13:18:24.860615] ============= Session Begin ============

T [Fri Sep 5 13:18:24.830472] ============= Session End ============

T [Fri Sep 5 13:18:24.830062] Commit phase: Time taken 0.98 ms, usr 0.00 ms,

sys 0.00 ms

T [Fri Sep 5 13:18:24.829085] ppf session [0xff0104b410000086] commit ... npi

nst 1

T [Fri Sep 5 13:18:24.827685] Verify phase: Time taken 2.04 ms, usr 0.00 ms,

sys 0.00 ms

T [Fri Sep 5 13:18:24.825388] ============= Session Begin ============

T [Fri Sep 5 12:32:51.364225] ============= Session End ============

T [Fri Sep 5 12:32:51.363748] Commit phase: Time taken 0.64 ms, usr 0.00 ms,

[snip]
```


ヒント：ファイルの中には大きなものがあるので、ブートフラッシュに送信して工ディタで調べる方が簡単です。

<#root>

```
module-1#
show system internal aclqos ?

asic            Asic information
brcm            Broadcam information
database        Database
event-history   Show various event logs of ACLQOS
mem-stats       Show memory allocation statistics of ACLQOS
prefix          External EPG prefixes
qos             QoS related information
range-resource  Zoning rules L4 destination port range resources
regions         Security TCAM priority regions
span            SPAN related information
zoning-rules    Show zoning rules

module-1#
show system internal aclqos event-history ?

errors        Show error logs of ACLQOS

msgs          Show various message logs of ACLQOS
ppf           Show ppf logs of ACLQOS
ppf-parse     Show ppf-parse logs of ACLQOS
prefix        Show prefix logs of ACLQOS
qos           Show qos logs of ACLQOS
qos-detail    Show detailed qos logs of ACLQOS
span          Show span logs of ACLQOS
span-detail   Show detailed span logs of ACLQOS

trace         Show trace logs of ACLQOS

trace-detail  Show detailed trace logs of ACLQOS

zoning-rules  Show detailed logs of ACLQOS
```

トラブルシューティングのヒント

トライブルシューティングに役立つヒントは、以下のとおりです。

- TCAM枯渇の問題が発生した場合は、UIまたはCLIで、問題のルールに関連するエラーを確認します。このエラーは次のように報告されます。

  <#root>

  Fault F1203 - Rule failed due to hardware programming error.

  1つのルールは、特定用途向け集積回路(ASIC)の複数のTCAMエントリを受け入れることができます。ASIC 上のエントリ数を表示するには、次のコマンドを入力します。

  <#root>

  fab1-leaf1#

  vsh_lc

  module-1#

  show platform internal ns table-health

  ```
  VLAN STATE curr usage: 0 - size: 4096
  QQ curr usage: 0 - size: 16384
  SEG STATE curr usage: 0 - size: 4096
  SRC TEP curr usage: 0 - size: 4096
  POLICY KEY curr usage: 0 - size: 1
  SRC VP curr usage: 0 - size: 4096

  SEC GRP curr usage: 43 - size: 4096
  ```

  注：この例では、43個のエントリがあります。この使用状況も、eqptCapacity クラスで APIC に報告されます。

- 複数の一致が見つかった場合は、TCAM ルックアップから低いほうのハードウェア インデックスが返されます。そのインデックスを確認するには、次のコマンドを入力します。

  <#root>

  show system internal aclqos zoning-rule

トラブルシューティングの際に、any-any-implicitルールによるドロップを確認できます。このルールは常に最下位に存在するため、このルールが存在しないことからパケットがドロップされていることになります。ルールが存在しない理由は、設定が誤っているか、ポリシー要素マネージャが正常にプログラムしていないためです。

- pcTags のスコープは、ローカルまたはグローバルのいずれかに設定されます。
    - System Reserved pcTag：このpcTagはシステム内部ルール(1 ~ 15)に使用されます。
    - グローバルスコープのpcTag：このpcTagは、共有サービス(16 ~ 16385)に使用されます。
    - ローカルスコープpcTag：このpcTagは、VRFごとにローカルに使用されます(16386 ~ 65535の範囲)。

トラブルシューティングする際は、値の長さを見るだけで、そのスコープがわかります。

ルールIDから契約名を取得

トラブルシューティングの場合、エンジニアはゾーニングのルールを確認することがよくあります。場合によっては、EPG/pcTagには多くのコントラクトがあり、トラブルシューティングが煩雑になることがあります。このセクションでは、スイッチのCLIで表示されるルールIDからEPGとpcTag間で使用されているコントラクトの名前を判別する方法の概要を説明します。

開始するには、次のようにします。

1. 具体的なコントラクト/ルールオブジェクトactrlRuleを必要に応じて照会し、プロパティで検索を絞り込みます。id値： rule-d

2. 正しいルールが見つかったら、DNの緑色の矢印をクリックして、actrlRuleオブジェクトの子を表示します。子供達が私たちの答えのあるところです。

actrlRule	?
action	permit
actrlCfgFailedBmp	
actrlCfgFailedTs	00:00:00:00.000
actrlCfgState	0
childAction	
dPcTag	16388
descr	
direction	uni-dir
dn	topology/pod-1/node-101/sys/actrl/scope-2719746/rule-2719746-s-49164-d-16388-f-38
fltId	38
id	4143
lcOwn	local
markDscp	unspecified
modTs	2016-01-08T19:44:02.267+00:00
monPolDn	uni/tn-common/monepg-default
name	
operSt	enabled
operStQual	
prio	fully_qual
qosGrp	unspecified
sPcTag	49164
scopeId	2719746
status	
type	tenant

ここでの子オブジェクトはactrlRsToEpgConnです。通常は、各EPGに1つずつ、合計2つのEPGを作成できます。このオブジェクトのDNは、コントラクトが適用される2つのEPGと、方向（プロバイダーまたはコンシューマー）および最も重要なコントラクトオブジェクト名を示します。

actrlRsToEpgConn	?
childAction	
dn	topology/pod-1/node-101/sys/actrl/scope-2719746/rule-2719746-s-49164-d-16388-f-38/rstoEpgConn-[cdef-[uni/tn-dpita-tenant/brc-dpita-ssh]/epgCont-[uni/tn-dpita-tenant/ap-dpita-AP/epg-dpita-EPG1]/fr-[uni/tn-dpita-tenant/brc-dpita-ssh/dirass/prov-[uni/tn-dpita-tenant/ap-dpita-AP/epg-dpita-EPG1]-any-no]/to-[uni/tn-dpita-tenant/brc-dpita-ssh/dirass/cons-[uni/tn-dpita-tenant/ap-dpita-AP/epg-dpita-EPG2]-any-no]]
forceResolve	no
lcOwn	local
modTs	2016-01-08T19:44:02.267+00:00
rType	mo
state	unformed
stateQual	none
status	
tCl	vzToEPg
tDn	cdef-[uni/tn-dpita-tenant/brc-dpita-ssh]/epgCont-[uni/tn-dpita-tenant/ap-dpita-AP/epg-dpita-EPG1]/fr-[uni/tn-dpita-tenant/brc-dpita-ssh/dirass/prov-[uni/tn-dpita-tenant/ap-dpita-AP/epg-dpita-EPG1]-any-no]/to-[uni/tn-dpita-tenant/brc-dpita-ssh/dirass/cons-[uni/tn-dpita-tenant/ap-dpita-AP/epg-dpita-EPG2]-any-no]
tType	mo

強調表示されているように、この場合のコントラクト名はbrc-dpita-sshです。

必要に応じて、vzBrCPに照会して適切な契約を見つけます。

vzBrCP	?
childAction	
configIssues	
descr	
dn	uni/tn-dpita-tenant/brc-dpita-ssh
lcOwn	local
modTs	2015-06-25T16:21:10.003+00:00
monPolDn	uni/tn-common/monepg-default
name	dpita-ssh
ownerKey	
ownerTag	
prio	unspecified
reevaluateAll	no
scope	context
status	
uid	15374

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。