# Security Policies

This chapter contains the following sections:

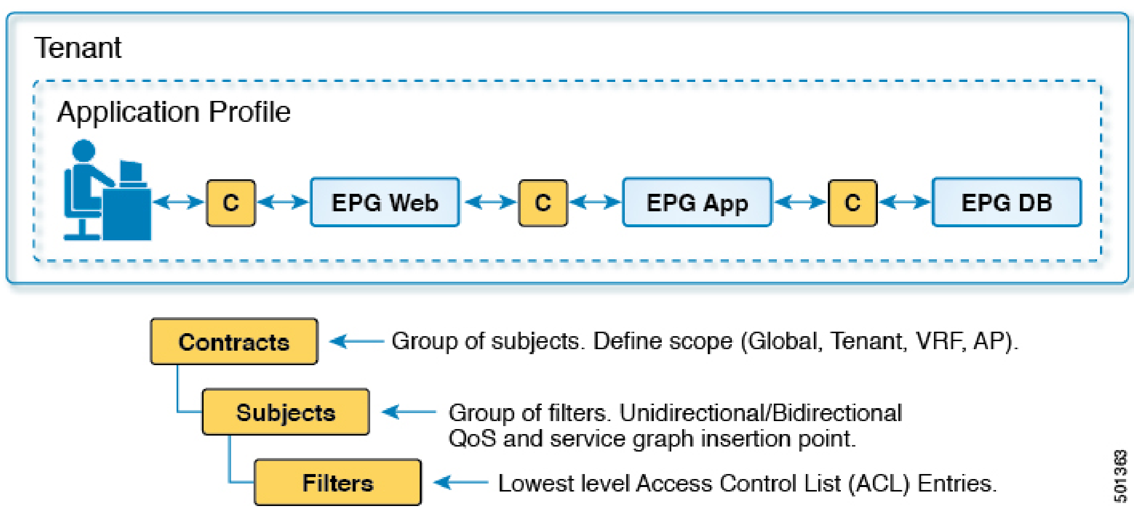


# ACI Fabric Network Access Security Policy Model (Contracts)

The ACI fabric security policy model is based on contracts. This approach addresses limitations of traditional access control lists (ACLs). Contracts contain the specifications for security policies that are enforced on traffic between endpoint groups.

The following figure shows the components of a contract.

***Figure 1: Contract Components***




EPG communications require a contract; EPG to EPG communication is not allowed without a contract. The APIC renders the entire policy model, including contracts and their associated EPGs, into the concrete model in each switch. Upon ingress, every packet entering the fabric is marked with the required policy details. Because contracts are required to select what types of traffic can pass between EPGs, contracts enforce security policies. While contracts satisfy the security requirements handled by access control lists (ACLs) in conventional network settings, they are a more flexible, manageable, and comprehensive security policy solution.

# Access Control List Limitations

Traditional access control lists (ACLs) have a number of limitations that the ACI fabric security model addresses. The traditional ACL is very tightly coupled with the network topology. They are typically configured per router or switch ingress and egress interface and are customized to that interface and the traffic that is expected to flow through those interfaces. Due to this customization, they often cannot be reused across interfaces, much less across routers or switches.

Traditional ACLs can be very complicated and cryptic because they contain lists of specific IP addresses, subnets, and protocols that are allowed as well as many that are specifically not allowed. This complexity means that they are difficult to maintain and often simply just grow as administrators are reluctant to remove any ACL rules for fear of creating a problem. Their complexity means that they are generally only deployed at specific demarcation points in the network such as the demarcation between the WAN and the enterprise or the WAN and the data center. In this case, the security benefits of ACLs are not exploited inside the enterprise or for traffic that is contained within the data center.

Another issue is the possible huge increase in the number of entries in a single ACL. Users often want to create an ACL that allows a set of sources to communicate with a set of destinations by using a set of protocols. In the worst case, if *N* sources are talking to *M* destinations using *K* protocols, there might be *N**M*K* lines in the ACL. The ACL must list each source that communicates with each destination for each protocol. It does not take many devices or protocols before the ACL gets very large.

The ACI fabric security model addresses these ACL issues. The ACI fabric security model directly expresses the intent of the administrator. Administrators use contract, filter, and label managed objects to specify how groups of endpoints are allowed to communicate. These managed objects are not tied to the topology of the network because they are not applied to a specific interface. They are simply rules that the network must enforce irrespective of where these groups of endpoints are connected. This topology independence means that these managed objects can easily be deployed and reused throughout the data center not just as specific demarcation points.

The ACI fabric security model uses the endpoint grouping construct directly so the idea of allowing groups of servers to communicate with one another is simple. A single rule can allow an arbitrary number of sources to communicate with an equally arbitrary number of destinations. This simplification dramatically improves their scale and maintainability which also means they are easier to use throughout the data center.

# Contracts Contain Security Policy Specifications

In the ACI security model, contracts contain the policies that govern the communication between EPGs. The contract specifies what can be communicated and the EPGs specify the source and destination of the communications. Contracts link EPGs, as shown below.

EPG 1 --------------- CONTRACT --------------- EPG 2

Endpoints in EPG 1 can communicate with endpoints in EPG 2 and vice versa if the contract allows it. This policy construct is very flexible. There can be many contracts between EPG 1 and EPG 2, there can be more than two EPGs that use a contract, and contracts can be reused across multiple sets of EPGs, and more.

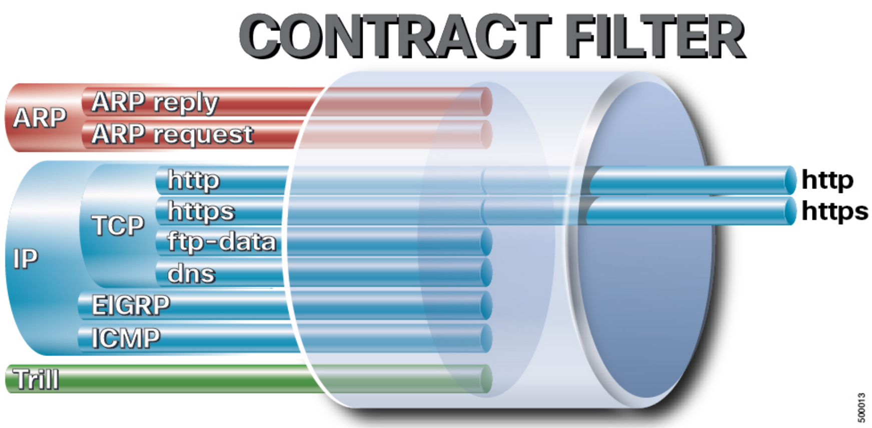
There is also directionality in the relationship between EPGs and contracts. EPGs can either provide or consume a contract. An EPG that provides a contract is typically a set of endpoints that provide a service to a set of client devices. The protocols used by that service are defined in the contract. An EPG that consumes a contract is typically a set of endpoints that are clients of that service. When the client endpoint (consumer) tries to connect to a server endpoint (provider), the contract checks to see if that connection is allowed. Unless otherwise specified, that contract would not allow a server to initiate a connection to a client. However, another contract between the EPGs could easily allow a connection in that direction.

This providing/consuming relationship is typically shown graphically with arrows between the EPGs and the contract. Note the direction of the arrows shown below.

EPG 1 <-------consumes-------- CONTRACT <-------provides-------- EPG 2

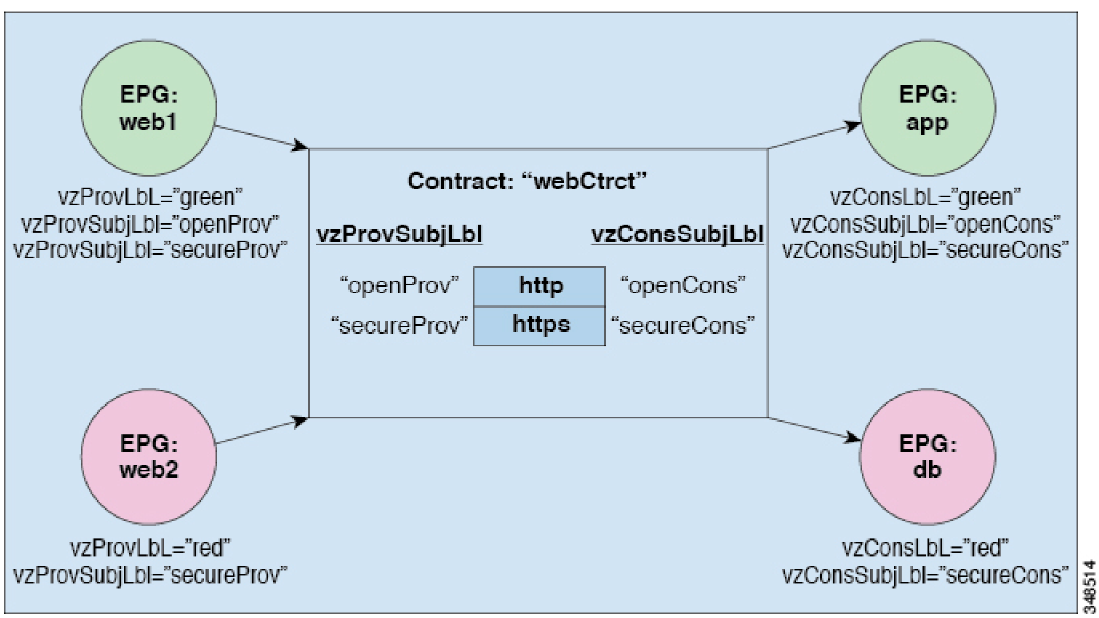
The contract is constructed in a hierarchical manner. It consists of one or more subjects, each subject contains one or more filters, and each filter can define one or more protocols.

*Figure 2: Contract Filters*




The following figure shows how contracts govern EPG communications.

*Figure 3: Contracts Determine EPG to EPG Communications*




For example, you may define a filter called HTTP that specifies TCP port 80 and port 8080 and another filter called HTTPS that specifies TCP port 443. You might then create a contract called webCtrct that has two sets of subjects. openProv and openCons are the subjects that contain the HTTP filter. secureProv and secureCons are the subjects that contain the HTTPS filter. This webCtrct contract can be used to allow both secure and non-secure web traffic between EPGs that provide the web service and EPGs that contain endpoints that want to consume that service.

These same constructs also apply for policies that govern virtual machine hypervisors. When an EPG is placed in a virtual machine manager (VMM) domain, the APIC downloads all of the policies that are associated with the EPG to the leaf switches with interfaces connecting to the VMM domain. For a full explanation of VMM domains, see the *Virtual Machine Manager Domains* chapter of *Application Centric Infrastructure Fundamentals*. When this policy is created, the APIC pushes it (pre-populates it) to a VMM domain that specifies which switches allow connectivity for the endpoints in the EPGs. The VMM domain defines the set of switches and ports that allow endpoints in an EPG to connect to. When an endpoint comes on-line, it is associated with the appropriate EPGs. When it sends a packet, the source EPG and destination EPG are derived from the packet and the policy defined by the corresponding contract is checked to see if the packet is allowed. If yes, the packet is forwarded. If no, the packet is dropped.

Contracts consist of 1 or more subjects. Each subject contains 1 or more filters. Each filter contains 1 or more entries. Each entry is equivalent to a line in an Access Control List (ACL) that is applied on the Leaf switch to which the endpoint within the endpoint group is attached.

In detail, contracts are comprised of the following items:

- Name—All contracts that are consumed by a tenant must have different names (including contracts created under the common tenant or the tenant itself).
- Subjects—A group of filters for a specific application or service.
- Filters—Used to classify traffic based upon layer 2 to layer 4 attributes (such as Ethernet type, protocol type, TCP flags and ports).
- Actions—Action to be taken on the filtered traffic. The following actions are supported:
  - Permit the traffic (regular contracts, only)
  - Mark the traffic (DSCP/CoS) (regular contracts, only)
  - Redirect the traffic (regular contracts, only, through a service graph)
  - Copy the traffic (regular contracts, only, through a service graph or SPAN)
  - Block the traffic (taboo contracts)

    With Cisco APIC Release 3.2(x) and switches with names that end in EX or FX, you can alternatively use a subject Deny action or Contract or Subject Exception in a standard contract to block traffic with specified patterns.
  - Log the traffic (taboo contracts and regular contracts)
- Aliases—(Optional) A changeable name for an object. Although the name of an object, once created, cannot be changed, the Alias is a property that can be changed.

Thus, the contract allows more complex actions than just allow or deny. The contract can specify that traffic that matches a given subject can be re-directed to a service, can be copied, or can have its QoS level modified. With pre-population of the access policy in the concrete model, endpoints can move, new ones can come on-line, and communication can occur even if the APIC is off-line or otherwise inaccessible. The APIC is removed from being a single point of failure for the network. Upon packet ingress to the ACI fabric, security policies are enforced by the concrete model running in the switch.

# Filter Entry Configuration

This section explains the following filter entry configuration options.

- Match Only Fragments
- Match DSCP
- TCP Flags
- Stateful
- Port Zero Entry

Each filter can contain one or more filter entries, which is located at **Tenant** > **Contract** > **Filters** > **Filter_name**, and the configuration location of each filter entry is at **Tenant** > **Contract** > **Filters** > **Filter_name** > **Filter_entry_name**.

## Match Only Fragments

The Match Only Fragments option is to match fragments with offset greater than 0 (all fragments except the first one).

The Match Only Fragments option is disabled by default. This means that the filter configurations by default are applied to all packets (including all fragments). Thus, by default all packets matched with the filter can be permitted, dropped, copied or redirected based on the contract action. When The Match Only Fragments option is enabled, the filter configurations are applied to all fragments except the first fragment.

**Note** TCP/UDP port information can only be checked in the first fragment.

Some examples are listed below:

- If a permit contract has an IP filter with “The Match Only Fragments” disabled (default), all IP packets including all fragments will be permitted.
- If a permit contract has an IP filter with “The Match Only Fragments” enabled, only IP fragments with offset greater than 0 (all IP fragments except the first one) will be permitted. Thus, the first fragment will be dropped by the implicit deny rule unless you have another permit contract.
- If a permit contract has a specific TCP port filter (such as destination TCP port 80) with “The Match Only Fragments” disabled (default) for a permit contract, all TCP traffic matched with the specific TCP port will be permitted. The fragments except the first one will be dropped by implicit deny rule unless you have another permit contract because TCP port information is in the first fragment only.
- The use of a specific TCP/UDP port filter with “The Match Only Fragments” enabled is not a valid configuration combination because TCP/UDP port information can only be checked in the first fragment whereas “The Match Only Fragments” is to match all fragments except the first one.

## Match DSCP

This option is to specify DSCP (Differentiated Services Code Point) value to match in the traffic in addition to EtherType, IP protocol, source port, and destination port. By using this option, different actions can be taken depending on which DSCP value is in the packet, even if other parameters, such as source EPG, destination EPG, and filter matching, are the same. This option is set, by default, to “Unspecified” (which in Cisco ACI is the equivalent of “Any” in classic IOS or NX-OS terminology). This requires leaf nodes with “EX” or “FX” onward.

## TCP Flags

This option is to specify the TCP flag values to match traffic in addition to EtherType, IP protocol, source port, and destination port. The available TCP flags are:

- Synchronize: SYN
- Established: ACK or RST
- Acknowledgement: ACK
- Reset: RST
- Finish: FIN

## Stateful

The Stateful option is to allow TCP packets from provider to consumer only if the ACK flag is set. This option is disabled by default. It is recommended to enable the Stateful option in TCP filter entries for better security except in those cases where Enable Policy Compression is required, because Policy compression cannot be applied if the Stateful option is enabled.

In order to let the consumer access a specific provider TCP port, the administrator must configure a consumer-side TCP port (the source port configuration in the contract filter) as wide range, to cover non-well-known source ports. The example below has two zoning rules: one rule to permit traffic from a consumer using an any-source TCP port to a provider with destination TCP port 80, and the other rule for the opposite direction. If a provider endpoint performs a SYN attack using the source TCP port 80 to a consumer endpoint, the traffic is automatically not dropped by the ACI fabric, because the traffic from the provider using source TCP port 80 to the consumer with an any destination TCP port is permitted by the contract.

When normal TCP packets from the provider to the consumer are permitted:

- Data packets (after a three-way handshake): These packets have the ACK bit set, so leaf nodes permit the packets.
- RST packet: RST packets also have ACK bit set, so leaf nodes permit RST packets.
- FIN packet: FIN packets with ACK bit set are permitted. FIN packets without ACK will be dropped. The handling of FIN packets without ACK differs based on the type of the operating system; therefore, it can be used for a FIN scan attack to determine the operating system. Dropping such packets can prevent such attacks.

The CLI output from the “show zoning-rule” command, is an example of a policy programmed on a leaf with the Stateful option enabled.

```
Pod1-Leaf1# show zoning-rule scope 2850817
+---------+--------+--------+----------+---------+---------+---------+---------+----------+-------------------+
| Rule ID | SrcEPG | DstEPG | FilterID |   Dir   |  operSt |  Scope  |         Name        |
  Action  |        Priority        |
+---------+--------+--------+----------+---------+---------+---------+---------+----------+-------------------+
|   4250  |    0    |   0    | implicit | uni-dir | enabled | 2850817 |                     |
 deny,log |    any_any_any(21)     |
|   4246  |    0    |   0    | implarp  | uni-dir | enabled | 2850817 |                     |
  permit  |   any_any_filter(17)   |
|   4208  |    0    |   15   | implicit | uni-dir | enabled | 2850817 |                     |
 deny,log | any_vrf_any_deny(22)   |
|   4247  |    0    | 32777  | implicit | uni-dir | enabled | 2850817 |                     |
  permit  |    any_dest_any(16)    |
|   4222  | 32774  | 32775  |    71    | uni-dir | enabled | 2850817 | tenant1:Contract1 |
```

```
  permit  |     fully_qual(7)      |
|   4244  | 32775  | 32774  |     69     | uni-dir | enabled | 2850817 | tenant1:Contract1 |
  permit  |     fully_qual(7)      |
+--------+--------+--------+---------+--------+--------+--------+--------------+---------+-----------------+
```

The lines are created by Contract1 between EPG Web and EPG App. The details of the filter entry information can be checked by using the command "show zoning-filter filter **FilterID**." The filter ID 71 used in the provider-to-consumer direction has TcpRules "ack."

```
Pod1-Leaf1# show zoning-filter filter 69
+----------+------+--------+-------------+------+-------------+----------+-------------+
| FilterId | Name | EtherT |    ArpOpc    | Prot | ApplyToFrag | Stateful |  SFromPort   |
 SToPort    | DFromPort | DToPort |  Prio |   Icmpv4T    |    Icmpv6T    | TcpRules |
+----------+------+--------+-------------+------+-------------+----------+-------------+
|    69    | 69_0 |   ip   | unspecified | tcp  |      no     |   yes    | unspecified |
unspecified |     22     |     22    | dport | unspecified | unspecified |           |
+----------+------+--------+-------------+------+-------------+----------+-------------+

Pod1-Leaf1# show zoning-filter filter 71
+----------+------+--------+-------------+------+-------------+----------+-----------+
| FilterId | Name | EtherT |    ArpOpc    | Prot | ApplyToFrag | Stateful | SFromPort |
SToPort |  DFromPort   |   DToPort    |  Prio |   Icmpv4T    |   Icmpv6T    | TcpRules |
+----------+------+--------+-------------+------+-------------+----------+-----------+
|    71    | 71_0 |   ip   | unspecified | tcp  |      no     |   yes    |     22    |
22    | unspecified | unspecified | flags | unspecified | unspecified |   ack    |
+----------+------+--------+-------------+------+-------------+----------+-----------+
```

The following list summarizes some of the key design considerations related to the use of the Stateful option:

- The Stateful option is applicable to TCP traffic only.
- The Stateful option just checks the ACK flag; it does not prevent an SYN + ACK attack from the provider, unlike a stateful firewall.
- Bidirectional rule compression cannot be applied if Stateful is enabled.

# Port Zero Entry

Each filter can contain one or more filter entries, which is located at **Tenant** > **Contract** > **Filters** > **Filter_name**.

Starting from APIC release 6.0(4), Port Zero Entry is introduced. The differences between a general filter entry and a Port Zero Entry are the followings:

- If port is set to "unspecified" or "0" in a general filter entry, it means the port range is "0-65535".
- Port Zero Entry is for a filter entry with port "0", which is mainly to deny such traffic because port "0" is defined as a reserved port by Internet Assigned Numbers Authority (IANA) and it is not supposed to be used.

Port Zero Entry has the following Direction options:

- Direction Both (default): source port "0" and destination port "0".
- Direction Destination: source port "0" and destination port "any"(0-65535).
- Direction Source: source port "any"(0-65535) and destination port "0".

**Note** A filter entry with either the source or the destination port “0” such as a filter with the source port “0” and the destination port “80” is not supported in either general filter entry or Port Zero Entry.

# Security Policy Enforcement

As traffic enters the leaf switch from the front panel interfaces, the packets are marked with the EPG of the source EPG. The leaf switch then performs a forwarding lookup on the packet destination IP address within the tenant space. A hit can result in any of the following scenarios:

1. A unicast (/32) hit provides the EPG of the destination endpoint and either the local interface or the remote leaf switch VTEP IP address where the destination endpoint is present.
2. A unicast hit of a subnet prefix (not /32) provides the EPG of the destination subnet prefix and either the local interface or the remote leaf switch VTEP IP address where the destination subnet prefix is present.
3. A multicast hit provides the local interfaces of local receivers and the outer destination IP address to use in the VXLAN encapsulation across the fabric and the EPG of the multicast group.

**Note** Multicast and external router subnets always result in a hit on the ingress leaf switch. Security policy enforcement occurs as soon as the destination EPG is known by the ingress leaf switch.

A miss result in the forwarding table causes the packet to be sent to the forwarding proxy in the spine switch. The forwarding proxy then performs a forwarding table lookup. If it is a miss, the packet is dropped. If it is a hit, the packet is sent to the egress leaf switch that contains the destination endpoint. Because the egress leaf switch knows the EPG of the destination, it performs the security policy enforcement. The egress leaf switch must also know the EPG of the packet source. The fabric header enables this process because it carries the EPG from the ingress leaf switch to the egress leaf switch. The spine switch preserves the original EPG in the packet when it performs the forwarding proxy function.

On the egress leaf switch, the source IP address, source VTEP, and source EPG information are stored in the local forwarding table through learning. Because most flows are bidirectional, a return packet populates the forwarding table on both sides of the flow, which enables the traffic to be ingress filtered in both directions.

# Multicast and EPG Security

Multicast traffic introduces an interesting problem. With unicast traffic, the destination EPG is clearly known from examining the packet’s destination. However, with multicast traffic, the destination is an abstract entity: the multicast group. Because the source of a packet is never a multicast address, the source EPG is determined in the same manner as in the previous unicast examples. The derivation of the destination group is where multicast differs.

Because multicast groups are somewhat independent of the network topology, static configuration of the (S, G) and (*, G) to group binding is acceptable. When the multicast group is placed in the forwarding table, the EPG that corresponds to the multicast group is also put in the forwarding table.

> **Note** This document refers to multicast stream as a multicast group.

The leaf switch always views the group that corresponds to the multicast stream as the destination EPG and never the source EPG. In the access control matrix shown previously, the row contents are invalid where the multicast EPG is the source. The traffic is sent to the multicast stream from either the source of the multicast stream or the destination that wants to join the multicast stream. Because the multicast stream must be in the forwarding table and there is no hierarchical addressing within the stream, multicast traffic is access controlled at the ingress fabric edge. As a result, IPv4 multicast is always enforced as ingress filtering.

The receiver of the multicast stream must first join the multicast stream before it receives traffic. When sending the IGMP Join request, the multicast receiver is actually the source of the IGMP packet. The destination is defined as the multicast group and the destination EPG is retrieved from the forwarding table. At the ingress point where the router receives the IGMP Join request, access control is applied. If the Join request is denied, the receiver does not receive any traffic from that particular multicast stream.

The policy enforcement for multicast EPGs occurs on the ingress by the leaf switch according to contract rules as described earlier. Also, the multicast group to EPG binding is pushed by the APIC to all leaf switches that contain the particular tenant (VRF).

## Taboos

While the normal processes for ensuring security still apply, the ACI policy model aids in assuring the integrity of whatever security practices are employed. In the ACI policy model approach, all communications must conform to these conditions:

- Communication is allowed only based on contracts, which are managed objects in the model. If there is no contract, inter-EPG communication is disabled by default.
- No direct access to the hardware; all interaction is managed through the policy model.

Taboo contracts can be used to deny specific traffic that is otherwise allowed by contracts. The traffic to be dropped matches a pattern (such as, any EPG, a specific EPG, or traffic matching a filter). Taboo rules are unidirectional, denying any matching traffic coming toward an EPG that provides the contract.

With Cisco APIC Release 3.2(x) and switches with names that end in EX or FX, you can alternatively use a subject Deny action or Contract or Subject Exception in a standard contract to block traffic with specified patterns.

# Enabling and Viewing ACL Contract and Deny Logs

## About ACL Contract Permit and Deny Logs

To log and/or monitor the traffic flow for a contract rule, you can enable and view the logging of packets or flows that were allowed to be sent because of contract permit rules or the logging of packets or flows that were dropped because of:

- Taboo contract deny rules
- Deny actions in contract subjects

- Contract or subject exceptions
- ACL contract permit in the ACI fabric is only supported on Nexus 9000 Series switches with names that end in EX or FX, and all later models. For example, N9K-C93180LC-EX or N9K-C9336C-FX.
- Deny logging in the ACI fabric is supported on all platforms.
- Using log directive on filters in management contracts is not supported. Setting the log directive will cause zoning-rule deployment failure.

For information on standard and taboo contracts and subjects, see *Cisco Application Centric Infrastructure Fundamentals* and *Cisco APIC Basic Configuration Guide*.

### EPG Data Included in ACL Permit and Deny Log Output

Up to Cisco APIC, Release 3.2(1), the ACL permit and deny logs did not identify the EPGs associated with the contracts being logged. In release 3.2(1) the source EPG and destination EPG are added to the output of ACI permit and deny logs. ACL permit and deny logs include the relevant EPGs with the following limitations:

- Depending on the position of the EPG in the network, EPG data may not be available for the logs.
- When configuration changes occur, log data may be out of date. In steady state, log data is accurate.

The most accurate EPG data in the permit and deny logs results when the logs are focussed on:

- Flows from EPG to EPG, where the ingress policy is installed at the ingress TOR and the egress policy is installed at the egress TOR.
- Flows from EPG to L3Out, where one policy is applied on the border leaf TOR and the other policy is applied on a non-BL TOR.

EPGs in the log output are not supported for uSeg EPGs or for EPGs used in shared services (including shared L3Outs).

# Enabling ACL Contract Permit and Deny Logging Using the GUI

The following steps show how to enable contract permit and deny logging using the GUI:

**Note** The tenant that contains the permit logging is the tenant that contains the VRF that the EPG is associated to. This will not necessarily be the same tenant as the EPG or its associated contracts.

### Procedure

**Step 1** On the menu bar, choose **Tenants** > **<tenant name>**.

**Step 2** In the **Navigation** pane, expand **Contracts**, right-click **Standard**, and choose **Create Contract**.

**Step 3** In the **Create Contract** dialog box, perform the following actions:

a) In the **Name** field, type the name for the contract.
b) In the **Scope** field, choose the scope for it (VRF, Tenant, or Global).

c) Optional. Set the target DSCP or QoS class to be applied to the contract.
d) Click the + icon to expand **Subjects**.

**Step 4** In the Create Contract Subject dialog box, perform the following actions:

**Step 5** Enter the name of the subject and an optional description.

**Step 6** Optional. From the drop-down list for the target DSCP, select the DSCP to be applied to the subject.

**Step 7** Leave **Apply Both Directions** checked, unless you want the contract to only be applied from the consumer to the provider, instead of in both directions.

**Step 8** Leave **Reverse Filter Ports** checked if you unchecked **Apply Both Directions** to swap the Layer 4 source and destination ports so that the rule is applied from the provider to the consumer.

**Step 9** Click the + icon to expand **Filters**.

**Step 10** In the **Name** drop-down list, choose an option; for example, click **arp**, **default**, **est**, or **icmp**, or choose a previously configured filter.

**Step 11** In the **Directives** drop-down list, click **log**.

**Step 12** (Optional) Change the Action to be taken with this subject to **Deny** (or leave the action to the default, **Permit**.

With Directive: log enabled, if the action for this subject is **Permit**, ACL permit logs track the flows and packets that are controlled by the subject and contract. If the action for this subject is **Deny**, ACL deny logs track the flows and packets.

**Step 13** (Optional) Set the priority for the subject.

**Step 14** Click **Update**.

**Step 15** Click **OK**.

**Step 16** Click **Submit**.
Logging is enabled for this contract.

# Enabling ACL Contract Permit Logging Using the NX-OS CLI

The following example shows how to enable Contract permit logging using the NX-OS CLI.

### Procedure

**Step 1** To enable logging of packets or flows that were allowed to be sent because of Contract permit rules, use the following commands:

```
configure
tenant <tenantName>
contract <contractName> type <permit>
subject <subject Name>
access-group <access-list> <in/out/both> log
```

**Example:**

For example:

```
apic1# configure
apic1(config)# tenant BDMode1
apic1(config-tenant)# contract Logicmp type permit
apic1(config-tenant-contract)# subject icmp
apic1(config-tenant-contract-subj)# access-group arp both log
```

**Step 2** To disable the permit logging use the **no** form of the access-group command; for example, use the `no access-group arp both log` command.

# Enabling ACL Contract Permit Logging Using the REST API

The following example shows you how to enable permit and deny logging using the REST API. This example configures ACL permit and deny logging for a contract with subjects that have Permit and Deny actions configured.

### Procedure

For this configuration, send a post with XML similar to the following example:

**Example:**

```
<vzBrCP dn="uni/tn-Tenant64/brc-C64" name="C64" scope="context">
    <vzSubj consMatchT="AtleastOne" name="HTTPSsbj" provMatchT="AtleastOne" revFltPorts="yes"
rn="subj-HTTPSsbj">
        <vzRsSubjFiltAtt action="permit" directives="log" forceResolve="yes" priorityOverride="default"

rn="rssubjFiltAtt-PerHTTPS" tDn="uni/tn-Tenant64/flt-PerHTTPS" tRn="flt-PerHTTPS"
tnVzFilterName="PerHTTPS"/>
    </vzSubj>
    <vzSubj consMatchT="AtleastOne" name="httpSbj" provMatchT="AtleastOne" revFltPorts="yes"
rn="subj-httpSbj">
       <vzRsSubjFiltAtt action="deny" directives="log" forceResolve="yes" priorityOverride="default"

rn="rssubjFiltAtt-httpFilter" tDn="uni/tn-Tenant64/flt-httpFilter" tRn="flt-httpFilter"
tnVzFilterName="httpFilter"/>
    </vzSubj>
    <vzSubj consMatchT="AtleastOne" name="subj64" provMatchT="AtleastOne" revFltPorts="yes"
rn="subj-subj64">
       <vzRsSubjFiltAtt action="permit" directives="log" forceResolve="yes" priorityOverride="default"

rn="rssubjFiltAtt-icmp" tDn="uni/tn-common/flt-icmp" tRn="flt-icmp" tnVzFilterName="icmp"/>
    </vzSubj>
</vzBrCP>
```

# Enabling Taboo Contract Deny Logging Using the GUI

The following steps show how to enable Taboo Contract deny logging using the GUI.

### Procedure

**Step 1** On the menu bar, choose **Tenants** > **<tenant name>**.

**Step 2** In the **Navigation** pane, expand **Contracts**.

**Step 3** Right-click **Taboos** and choose **Create Taboo Contract**.

**Step 4** In the Create Taboo Contract dialog box, perform the following actions to specify the Taboo contract:

a) In the **Name** field, type the name for the contract.
b) Optional. In the **Description** field, type a description of the Taboo contract.
c) Click the + icon to expand **Subjects**.

**Step 5** In the **Create Taboo Contract Subject** dialog box, perform the following actions:

a) In the Specify Identity of Subject area, type a name and optional description.
b) Click the + icon to expand **Filters**.
c) From the **Name** drop-down list, choose one of the default values, such as**<tenant_name>/arp**, **<tenant_name>/default**, **<tenant_name>/est**, **<tenant_name>/icmp**, choose a previously created filter, or**Create Filter**.

**Note** If you chose **Create Filter**, in the Specify Filter Identity Area, perform the following actions to specify criteria for the ACL Deny rule:

**a.** Type a name and optional description.

**b.** Expand **Entries**, type a name for the rule, and choose the criteria to define the traffic you want to deny.

**c.** In theDirectives drop-down list, choose **log**.

**d.** Click **Update**.

**e.** Click **OK**.

**Step 6** Click **Submit**.
Logging is enabled for this Taboo contract.

# Enabling Taboo Contract Deny Logging Using the NX-OS CLI

The following example shows how to enable Taboo Contract deny logging using the NX-OS CLI.

### Procedure

**Step 1** To enable logging of packets or flows dropped because of Taboo Contract deny rules, use the following commands:

```
configure
tenant <tenantName>
contract <contractName> type <deny>
subject <subject Name>
access-group <access-list> <both> log
```

**Example:**

For example:

```
apic1# configure
apic1(config)# tenant BDMode1
apic1(config-tenant)# contract dropFTP type deny
apic1(config-tenant-contract)# subject dropftp
apic1(config-tenant-contract-subj)# access-group ftp both log
```

**Step 2** To disable the deny logging use the **no** form of the access-group command; for example, use the `no access-group https both log` command.

# Enabling Taboo Contract Deny Logging Using the REST API

The following example shows you how to enable Taboo Contract deny logging using the REST API.

### Procedure

To configure taboo contract deny logging, send a post with XML similar to the following example.

**Example:**

```
<vzTaboo dn="uni/tn-Tenant64/taboo-TCtrctPrefix" name="TCtrctPrefix" scope="context">
    <vzTSubj name="PrefSubj" rn="tsubj-PrefSubj"">
        <vzRsDenyRule directives="log" forceResolve="yes" rn="rsdenyRule-default" tCl="vzFilter"
tDn="uni/tn-common/flt-default" tRn="flt-default"/>
    </vzTSubj>
</vzTaboo>
```

# Viewing ACL Permit and Deny Logs Using the GUI

The following steps show how to view ACL permit and deny logs (if they are enabled) for traffic flows, using the GUI:

### Procedure

**Step 1** On the menu bar, choose **Tenants** > **<tenant name>**.

**Step 2** In the **Navigation** pane, click on **Tenant <tenant name>**.

**Step 3** In the **Tenants <tenant name> Work** pane, click the **Operational** tab.

**Step 4** Under the **Operational** tab, click the **Flows** tab.
Under the **Flows** tab, click one of the tabs to view log data for Layer 2 permit logs (**L2 Permit**) Layer 3 permit logs (**L3 Permit**, Layer 2 deny logs (**L2 Drop**), or Layer 3 deny logs (**L3 Drop**). On each tab, you can view ACL logging data, if traffic is flowing. The data points differ according to the log type and ACL rule; for example, the following data points are included for **L3 Permit** and **L3 Deny** logs:

- VRF
- Alias
- Source IP address
- Destination IP address
- Protocol
- Source port
- Destination port
- Source MAC address
- Destination MAC address

- Node
- Source interface
- VRF Encap
- Source EPG
- Destination EPG
- Source PC Tag
- Destination PC Tag

**Note** You can also use the **Packets** tab (next to the **Flows** tab) to access ACL logs for groups of packets (up to 10) with the same signature, source and destination. You can see what type of packets are being sent and which are being dropped.

# Viewing ACL Permit and Deny Logs Using the REST API

The following example shows how to view Layer 2 deny log data for traffic flows, using the REST API. You can send queries using the following MOs:

- acllogDropL2Flow
- acllogPermitL2Flow
- acllogDropL3Flow
- acllogPermitL3Flow
- acllogDropL2Pkt
- acllogPermitL2Pkt
- acllogDropL3Pkt
- acllogPermitL3Pkt

### Before you begin

You must enable permit or deny logging, before you can view ACL contract permit and deny log data.

## Procedure

To view Layer 3 drop log data, send the following query using the REST API:

```
GET https://apic-ip-address/api/class/acllogDropL3Flow
```

**Example:**

The following example shows sample output:


```
<?xml version="1.0" encoding="UTF-8"?>
<imdata totalCount="2">
```

```
    <acllogPermitL3Flow childAction="" dn="topology/pod-1/node-101/ndbgs/acllog/tn-common/ctx-inb

/permitl3flow-spctag-333-dpctag-444-sepgname-unknown-depgname-unknown-sip-[100:c000:a00:700:b00:0:f00:0]


-dip-[19.0.2.10]-proto-udp-sport-17459-dport-8721-smac-00:00:15:00:00:28-dmac-00:00:12:00:00:25-sintf-

        [port-channel5]-vrfencap-VXLAN: 2097153" dstEpgName="unknown" dstIp="19.0.2.10"
dstMacAddr="00:00:12:00:00:25"
        dstPcTag="444" dstPort="8721" lcOwn="local" modTs="never" monPolDn="" protocol="udp"
srcEpgName="unknown"
        srcIntf="port-channel5" srcIp="100:c000:a00:700:b00:0:f00:0" srcMacAddr="00:00:15:00:00:28"
 srcPcTag="333"
        srcPort="17459" status="" vrfEncap="VXLAN: 2097153"/>
    <acllogPermitL3Flow childAction="" dn="topology/pod-1/node-102/ndbgs/acllog/tn-common/ctx-inb

/permitl3flow-spctag-333-dpctag-444-sepgname-unknown-depgname-unknown-sip-[100:c000:a00:700:b00:0:f00:0]-dip-


[19.0.2.10]-proto-udp-sport-17459-dport-8721-smac-00:00:15:00:00:28-dmac-00:00:12:00:00:25-sintf-
        [port-channel5]-vrfencap-VXLAN: 2097153" dstEpgName="unknown" dstIp="19.0.2.10"
dstMacAddr="00:00:12:00:00:25"
        dstPcTag="444" dstPort="8721" lcOwn="local" modTs="never" monPolDn="" protocol="udp"
srcEpgName="unknown"
        srcIntf="port-channel5" srcIp="100:c000:a00:700:b00:0:f00:0" srcMacAddr="00:00:15:00:00:28"
 srcPcTag="333"
        srcPort="17459" status="" vrfEncap="VXLAN: 2097153"/>
</imdata>
```


# Viewing ACL Permit and Deny Logs Using the NX-OS CLI

The following steps show how to view ACL log details using the NX-OS-style CLI **show acllog** command.

The syntax for the Layer 3 command is **show acllog {permit | deny} l3 {pkt | flow} tenant <tenant_name> vrf <vrf_name> srcip <source_ip> dstip <destination_ip> srcport <source_port> dstport <destination_port> protocol <protocol> srcintf <source_interface> start-time <startTime> end-time <endTime> detail**

The syntax for the Layer 2 command is **show acllog {permit | deny} l2 {flow | pkt} tenant <tenant_name> vrf <VRF_name> srcintf <source_interface> vlan <VLAN_number> detail**

**Note** The full syntax of the **show acllog** command is only available on Generation 2 Cisco Nexus 9000 series switches (with names that end in EX or FX or later, such as N9K-C93180LC-EX) and Cisco APIC Release 3.2 or later. With Generation 1 switches (with names that do not end in EX or FX) or Cisco APIC releases before 3.2, the available syntax is as above.

In Cisco APIC 3.2 and later, additional keywords are added to both versions of the command, with the **detail** keyword:**[dstEpgName <destination_EPG_name>| dstmac <destination_MAC_address> | dstpctag <destination_PCTag> | srcEpgName <source_EPG_name> | srcmac <source_MAC_address> | srcpctag <source_PCTag>]**

## Procedure

**Step 1** The following example shows how to use the **show acllog drop l3 flow tenant common vrf default detail** command to display detailed information about Layer 3 deny logs for the common tenant:

**Example:**

```
apic1# show acllog deny l3 flow tenant common vrf default detail
SrcPcTag  : 49153
DstPcTag  : 32773
SrcEPG    : uni/tn-TSW_Tenant0/ap-tsw0AP0/epg-tsw0ctx0BD0epg6
DstEPG    : uni/tn-TSW_Tenant0/ap-tsw0AP0/epg-tsw0ctx0BD0epg5
SrcIp     : 16.0.2.10
DstIp     : 19.0.2.10
Protocol  : udp
SrcPort   : 17459
DstPort   : 8721
SrcMAC    : 00:00:15:00:00:28
DstMAC    : 00:00:12:00:00:25
Node      : 101
SrcIntf   : port-channel5
VrfEncap  : VXLAN: 2097153
```

This example shows the output on Generation 2 switches, with Cisco APIC Release 3.2 or later.

**Step 2** The following example shows how to use the **show acllog deny l2 flow tenant common vrf tsw0connctx0 detail** command to display detailed information about Layer 3 deny logs for the common tenant:

**Example:**

```
apic1# show acllog deny l2 flow tenant common vrf tsw0connctx0 detail
SrcPcTag DstPcTag   SrcEPG           DstEPG          SrcMAC            DstMAC         Node   SrcIntf
   vlan
--------- --------- ----------------  --------------  -------------------  ------------------   -----
   ---------  ------
 32773    49153   uni/tn-TSW       uni/tn-TSW  00:00:11:00:00:11  11:00:32:00:00:33   101     port-
     2
                 _Tenant0/ap-     _Tenant0/ap-                                                channel8

                  tsw0AP0/epg-     tsw0AP0/epg-
                  tsw0ctx0BD0epg5  tsw0ctx0BD0epg6
```

This example shows the output on Generation 2 switches, with Cisco APIC Release 3.2 or later.

**Step 3** The following example shows how to use the **show acllog permit l3 pkt tenant <tenant name> vrf <vrf name> [detail]** command to display detailed information about the common VRF ACL Layer 3 permit packets that were sent:

```
apic1# show acllog permit l3 pkt tenant common vrf default detail acllog permit l3 packets detail:
srcIp      : 10.2.0.19
dstIp      : 10.2.0.16
protocol   : udp
srcPort    : 13124
dstPort    : 4386
srcIntf    : port-channel5
vrfEncap   : VXLAN: 2097153
pktLen     : 112
srcMacAddr : 00:00:15:00:00:28
dstMacAddr : 00:00:12:00:00:25
timeStamp  : 2015-03-17T21:31:14.383+00:00
```

This example shows the output on Generation 1 switches, or with Cisco APIC releases before 3.2.

**Step 4** The following example shows how to use the **show acllog permit l2 pkt tenant <tenant name> vrf <vrf name> srcintf <s interface>** command to view information about default VRF Layer 2 packets sent from interface port-channel15:

```
apic1# show acllog permit l2 pkt tenant common vrf default srcintf port-channel5
acllog permit L2 Packets
      Node          srcIntf       pktLen     timeStamp
 --------------  --------------  --------  --------------
                 port-channel5      1      2015-03-17T21:
                                           31:14.383+00:00
```

This example shows the output on Generation 1 switches, or with Cisco APIC releases before 3.2.