Configuring Security

Enabling TACACS+, RADIUS, and LDAP

Overview

This article provides step by step instructions on how to enable RADIUS, TACACS+, and LDAP users access the APIC. It assumes the reader is thoroughly familiar with the Cisco Application Centric Infrastructure Fundamentals manual, especially the User Access, Authentication, and Accounting chapter.

Note Remote users for AAA Authentication with shell:domains=all/read-all/ will not be able to access Leaf switches and Spine switches in the fabric for security purposes. This pertains to all version up to 4.0(1h).

Guidelines

When configuring an external authentication server to access the Cisco APIC, follow these guidelines:

- Whenever a .
- While configuring .
- By definition, .
- This configuration task is applicable for .

- The user must configure .
- The autonomous system feature can only be .

Configuring APIC for TACACS+ Using the REST API

- The Cisco Application Centric Infrastructure (ACI) fabric must be installed, Application Policy Infrastructure Controllers (APICs) must be online, and the APIC cluster must be formed and healthy.
- The TACACS+ server host name or IP address, port, and key must be available.
- The APIC management endpoint group must be available.

Step 1 Configure the TACACS+ Provider by sending a POST request with XML such as the following example:

Example:

```
<aaaTacacsPlusProvider timeout="5" retries="1" port="49" name="192.168.200.1" monitoringUser="test"
 monitorServer="disabled"
dn="uni/userext/tacacsext/tacacsplusprovider-192.168.200.1" authProtocol="pap"/>
```

Step 2 Configure the TACACS+ Provider Group by sending a POST request with XML such as the following example:

Example:

```
<aaaTacacsPlusProviderGroup name="TENANT64_TACACS_provGrp"
dn="uni/userext/tacacsext/tacacsplusprovidergroup-TENANT64_TACACS_provGrp"/>
```

Step 3 Configure the TACACS+ Login Domain by sending a POST request with XML such as the following example:

Example:

```
<aaaLoginDomain name="TENANT64_TACACS_LoginDom" dn="uni/userext/logindomain-TENANT64_TACACS_LoginDom"/>
```

Example

The entire configuration can be sent in one POST request, with XML such as this example:

```
<aaaTacacsPlusProvider timeout="5" retries="1" port="49"
name="192.168.200.1" monitoringUser="test" monitorServer="disabled"
dn="uni/userext/tacacsext/tacacsplusprovider-192.168.200.1" authProtocol="pap"/>
<aaaTacacsPlusProviderGroup name="TENANT64_TACACS_provGrp"
dn="uni/userext/tacacsext/tacacsplusprovidergroup-TENANT64_TACACS_provGrp"/>
<aaaLoginDomain name="TENANT64_TACACS_LoginDom"
dn="uni/userext/logindomain-TENANT64_TACACS_LoginDom"/>
```

Configuring APIC for RADIUS Using the REST API

Before you begin

- The ACI fabric must be installed, Application Policy Infrastructure Controllers (APICs) must be online, and the APIC cluster must be formed and healthy.
- The RADIUS server host name or IP address, port, authorization protocol, and key must be available.

- The APIC management endpoint group must be available.

Step 1 Configure the RADIUS Provider by sending a POST request with XML such as the following example:

Example:

```
<aaaRadiusProvider timeout="5" retries="1" name="TENANT64_RADIUS-host.com"
monitoringUser="test" monitorServer="disabled"
dn="uni/userext/radiusext/radiusprovider-TENANT64_RADIUS-host.com" authProtocol="pap" authPort="1812"/>
```

Step 2 Configure the RADIUS Provider Group by sending a POST request with XML such as the following example:

Example:

```
<aaaRadiusProviderGroup name="TENANT64_RADIUS_provGrp"
dn="uni/userext/radiusext/radiusprovidergroup-TENANT64_RADIUS_provGrp"/>
```

Step 3 Configure the RADIUS Login Domain by sending a POST request with XML such as the following example:

Example:

```
<aaaLoginDomain name="TENANT64_RADIUSLoginDom"
dn="uni/userext/logindomain-TENANT64_RADIUSLoginDom"/>
```

Example

The entire configuration can be sent as one POST request, with XML such as this example:

```
<aaaRadiusProvider
timeout="5" retries="1" name="TENANT64_RADIUS-host.com" monitoringUser="test"
monitorServer="disabled"
dn="uni/userext/radiusext/radiusprovider-TENANT64_RADIUS-host.com" authProtocol="pap"
authPort="1812"/>
<aaaRadiusProviderGroup
name="TENANT64_RADIUS_provGrp"
dn="uni/userext/radiusext/radiusprovidergroup-TENANT64_RADIUS_provGrp"/>
<aaaLoginDomain
name="TENANT64_RADIUSLoginDom" dn="uni/userext/logindomain-TENANT64_RADIUSLoginDom"/>
```

Configuring APIC for LDAP Using the REST API

Before you begin

- The Cisco Application Centric Infrastructure (ACI) fabric must be installed, Application Policy Infrastructure Controllers (APICs) must be online, and the APIC cluster must be formed and healthy.
- The LDAP server host name or IP address, port, bind DN, Base DN, and password must be available.
- The APIC management endpoint group must be available.

Step 1 Configure the LDAP Provider by sending a POST request with XML such as the following example:

Example:

```
<aaaLdapProvider timeout="30" rootdn="" retries="1" port="389" name="TENANT64_LDAP-host.com"
monitoringUser="test" monitorServer="disabled" filter="cn=$userid" enableSSL="yes"
dn="uni/userext/ldapext/ldapprovider-TENANT64_LDAP-host.com" descr="" basedn=""
attribute="CiscoAVPair" SSLValidationLevel="strict"/>
```

Step 2 Configure the LDAP Provider Group by sending a POST request with XML such as the following example:

Example:

```
<aaaLdapProviderGroup name="TENANT64_LDAP-ProvGrp"
dn="uni/userext/ldapext/ldapprovidergroup-TENANT64_LDAP-ProvGrp"/>
```

Step 3 Configure the LDAP Login Domain by sending a POST request with XML such as the following example:

Example:

```
<aaaDomainAuth realm="ldap" providerGroup="TENANT64_LDAP-ProvGrp"
dn="uni/userext/logindomain-TENANT64_LDAPLoginDom/domainauth"/>
```

Example

The entire configuration can be sent in one POST request, with XML such as the following example:

```
<aaaLdapProvider
timeout="30" rootdn="" retries="1" port="389" name="TENANT64_LDAP-host.com"
monitoringUser="test" monitorServer="disabled" filter="cn=$userid" enableSSL="yes"
dn="uni/userext/ldapext/ldapprovider-TENANT64_LDAP-host.com" descr="" basedn=""
attribute="CiscoAVPair" SSLValidationLevel="strict"/>
<aaaLdapProviderGroup
name="TENANT64_LDAP-ProvGrp"dn="uni/userext/ldapext/ldapprovidergroup-TENANT64_LDAP-ProvGrp"/>
<aaaDomainAuth
realm="ldap" providerGroup="TENANT64_LDAP-ProvGrp"
dn="uni/userext/logindomain-TENANT64_LDAPLoginDom/domainauth"/>
```

Configuring FIPS

About Federal Information Processing Standards (FIPS)

The Federal Information Processing Standards (FIPS) Publication 140-2, Security Requirements for Cryptographic Modules, details the U.S. government requirements for cryptographic modules. FIPS 140-2 specifies that a cryptographic module should be a set of hardware, software, firmware, or some combination that implements cryptographic functions or processes, including cryptographic algorithms and, optionally, key generation, and is contained within a defined cryptographic boundary.

FIPS specifies certain cryptographic algorithms as secure, and it also identifies which algorithms should be used if a cryptographic module is to be called FIPS compliant.

Guidelines and Limitations for FIPS

The following guidelines and limitations apply to FIPS:

- When FIPS is enabled, FIPS is applied across the Cisco Application Policy Infrastructure Controller (APIC).

- When FIPS is enabled, you must disable FIPS before you downgrade the Cisco APIC to a release that does not support FIPS.
- Make your passwords a minimum of eight characters in length.
- Disable Telnet. Log in using only SSH.
- Delete all SSH Server RSA1 keypairs.
- Secure Shell (SSH) and SNMP are supported.
- Disable SNMP v1 and v2. Any existing user accounts on the switch that have been configured for SNMPv3 should be configured only with SHA for authentication and AES for privacy.
- Disable remote authentication through RADIUS/TACACS+. Only local and LDAP users can be authenticated.
- After enabling FIPS on the Cisco APIC, reload the dual supervisor spine switches twice for FIPS to take effect.
- On a dual supervisor spine switch that has FIPS enabled, if a supervisor is replaced, then the spine switch must be reloaded twice for FIPS to take effect on the new supervisor.
- Starting with the 2.3(1) release, FIPS can be configured at the switch level.
- Starting with the 3.1(1) release, when FIPs is enabled, NTP will operate in FIPS mode, Under FIPS mode NTP supports authentication with HMAC-SHA1 and no authentication.

Configuring FIPS for Cisco APIC Using REST API

When FIPS is enabled, it is applied across Cisco APIC.

Configure FIPS for all tenants.

Example:

```
https://apic1.cisco.com/api/node/mo/uni/userext.xml
<aaaFabricSec fipsMode="enable" />
```

Note You must reboot to complete the configuration. Anytime you change the mode, you must reboot to complete the configuration.

Configuring Fabric Secure Mode

Fabric Secure Mode

Fabric secure mode prevents parties with physical access to the fabric equipment from adding a switch or APIC controller to the fabric without manual authorization by an administrator. Starting with release 1.2(1x), the firmware checks that switches and controllers in the fabric have valid serial numbers associated with a

valid Cisco digitally signed certificate. This validation is performed upon upgrade to this release or during an initial installation of the fabric. The default setting for this feature is permissive mode; an existing fabric continues to run as it has after an upgrade to release 1.2(1) or later. An administrator with fabric-wide access rights must enable strict mode. The following table summarizes the two modes of operation:

Permissive Mode (default)	Strict Mode
Allows an existing fabric to operate normally even though one or more switches have an invalid certificate.	Only switches with a valid Cisco serial number and SSL certificate are allowed.
Does not enforce serial number based authorization .	Enforces serial number authorization.
Allows auto-discovered controllers and switches to join the fabric without enforcing serial number authorization.	Requires an administrator to manually authorize controllers and switches to join the fabric.

Configuring Fabric Secure Mode Using the REST API

To manage Secure Fabric Mode using the REST API, perform the following steps:

Step 1 To enable strict mode, send a POST request with XML such as the following example:

Example:

```
POST https://apic-ip-address/api/node/mo/uni.xml?
  <pkiFabricCommunicationEp mode="strict"/>
```

Step 2 To enable permissive mode, send a POST request with XML such as the following example:

Example:

```
POST https://apic-ip-address/api/node/mo/uni.xml?
  <pkiFabricCommunicationEp mode="permissive"/>
```

Step 3 To authorize a controller, send a POST request with XML such as the following example:

Example:

```
POST https://apic-ip-address/api/mo/uni/controller.xml?
  <fabricNodeIdentPol>
      <fabricCtrlrIdentP serial="TEP-1-1"/>
  </fabricNodeIdentPol>
```

Step 4 To reject a controller, send a POST request with XML such as the following example:

Example:

```
POST https://apic-ip-address/api/mo/uni/controller.xml?
  <fabricNodeIdentPol>
      <fabricCtrlrIdentP serial="FCH1750V025" reject="yes"/>
  </fabricNodeIdentPol>
```

Enabling RBAC

Access Rights Workflow Dependencies

The Cisco Application Centric Infrastructure (ACI) RBAC rules enable or restrict access to some or all of the fabric. For example, in order to configure a leaf switch for bare metal server access, the logged in administrator must have rights to the `infra` domain. By default, a tenant administrator does not have rights to the `infra` domain. In this case, a tenant administrator who plans to use a bare metal server connected to a leaf switch could not complete all the necessary steps to do so. The tenant administrator would have to coordinate with a fabric administrator who has rights to the `infra` domain. The fabric administrator would set up the switch configuration policies that the tenant administrator would use to deploy an application policy that uses the bare metal server attached to an ACI leaf switch.

AAA RBAC Roles and Privileges

The Application Policy Infrastructure Controller (APIC) provides the following AAA roles and privileges:

Note For each of the defined roles in Cisco APIC, the *APIC Roles and Privileges Matrix* shows which managed object classes can be written and which can be read. The matrix can be found at this URL: https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/apicroles/roles.html

Role	Privilege	Description
aaa	aaa	Used for configuring authentication, authorization, accounting, and import/export policies.
admin	admin	Provides full access to all of the features of the fabric. The admin privilege can be considered to be a union of all other privileges.

Role: access-admin	
Privilege	**Description**
access-connectivity-l1	Used for Layer 1 configuration under infra. Example: selectors and port Layer 1 policy configurations.
access-connectivity-l2	Used for Layer 2 configuration under infra. Example: encap configurations on selectors, and attachable entity.
access-connectivity-l3	Used for Layer 3 configuration under infra and static route configurations under a tenant's L3Out.
access-connectivity-mgmt	Used for management infra policies.
access-connectivity-util	Used for tenant ERSPAN policies.

Role: access-admin	
Privilege	**Description**
access-equipment	Used for access port configuration.
access-protocol-l1	Used for Layer 1 protocol configurations under infra.
access-protocol-l2	Used for Layer 2 protocol configurations under infra.
access-protocol-l3	Used for Layer 3 protocol configurations under infra.
access-protocol-mgmt	Used for fabric-wide policies for NTP, SNMP, DNS, and image management.
access-protocol-ops	Used for operations-related access policies such as cluster policy and firmware policies.
access-qos	Used for changing CoPP and QoS-related policies.

Role:fabric-admin	
Privilege	**Description**
fabric-connectivity-l1	Used for Layer 1 configuration under the fabric. Example: selectors and port Layer 1 policy and vPC protection.
fabric-connectivity-l2	Used in firmware and deployment policies for raising warnings for estimating policy deployment impact.
fabric-connectivity-l3	Used for Layer 3 configuration under the fabric. Example: Fabric IPv4, IPv6, and MAC protection groups.
fabric-connectivity-mgmt	Used for atomic counter and diagnostic policies on leaf switches and spine switches.
fabric-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
fabric-equipment	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
fabric-protocol-l1	Used for Layer 1 protocol configurations under the fabric.
fabric-protocol-l2	Used for Layer 2 protocol configurations under the fabric.
fabric-protocol-l3	Used for Layer 3 protocol configurations under the fabric.
fabric-protocol-mgmt	Used for fabric-wide policies for NTP, SNMP, DNS, and image management.
fabric-protocol-ops	Used for ERSPAN and health score policies.
fabric-protocol-util	Used for firmware management traceroute and endpoint tracking policies.

Role:fabric-admin	
Privilege	**Description**
tenant-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
tenant-connectivity-l2	Used for Layer 2 connectivity changes, including bridge domains and subnets.
tenant-connectivity-l3	Used for Layer 3 connectivity changes, including VRFs.
tenant-protocol-ops	Used for tenant traceroute policies.

Role	Privilege	Description
nw-svc-admin	nw-svc-device	Used for managing Layer 4 to Layer 7 service devices.
	nw-svc-devshare	Used for managing shared Layer 4 to Layer 7 service devices.
	nw-svc-policy	Used for managing Layer 4 to Layer 7 network service orchestration.
nw-svc-params	nw-svc-params	Used for managing Layer 4 to Layer 7 service policies.

Role: ops	
Privilege	**Description**
ops	Used for viewing the policies configured including troubleshooting policies. **Note** The **ops** role cannot be used for creating new monitoring and troubleshooting policies. Those policies need to be created by using the **admin** privilege, just like any other configurations in the Cisco APIC.

Role: read-all	
Privilege	**Description**
access-connectivity-l1	Used for Layer 1 configuration under infra. Example: selectors and port Layer 1 policy configurations.
access-connectivity-l2	Used for Layer 2 configuration under infra. Example: Encap configurations on selectors, and attachable entity.
access-connectivity-l3	Used for Layer 3 configuration under infra and static route configurations under a tenant's L3Out.

Role: read-all	
Privilege	**Description**
access-connectivity-mgmt	Used for management infra policies.
access-connectivity-util	Used for tenant ERSPAN policies.
access-equipment	Used for access port configuration.
access-protocol-l1	Used for Layer 1 protocol configurations under infra.
access-protocol-l2	Used for Layer 2 protocol configurations under infra.
access-protocol-l3	Used for Layer 3 protocol configurations under infra.
access-protocol-mgmt	Used for fabric-wide policies for NTP, SNMP, DNS, and image management.
access-protocol-ops	Used for operations-related access policies such as cluster policy and firmware policies.
access-qos	Used for changing CoPP and QoS-related policies.
fabric-connectivity-l1	Used for Layer 1 configuration under the fabric. Example: selectors and port Layer 1 policy and vPC protection.
fabric-connectivity-l2	Used in firmware and deployment policies for raising warnings for estimating policy deployment impact.
fabric-connectivity-l3	Used for Layer 3 configuration under the fabric. Example: Fabric IPv4, IPv6, and MAC protection groups.
fabric-protocol-l1	Used for Layer 1 protocol configurations under the fabric.
fabric-protocol-l2	Used for Layer 2 protocol configurations under the fabric.
fabric-protocol-l3	Used for Layer 3 protocol configurations under the fabric.
nw-svc-device	Used for managing Layer 4 to Layer 7 service devices.
nw-svc-devshare	Used for managing shared Layer 4 to Layer 7 service devices.
nw-svc-params	Used for managing Layer 4 to Layer 7 service policies.
nw-svc-policy	Used for managing Layer 4 to Layer 7 network service orchestration.
ops	Used for viewing the policies configured including troubleshooting policies. **Note** The **ops** role cannot be used for creating new monitoring and troubleshooting policies. Those policies need to be created by using the **admin** privilege, just like any other configurations in the Cisco APIC.

Role: read-all	
Privilege	**Description**
tenant-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
tenant-connectivity-l2	Used for Layer 2 connectivity changes, including bridge domains and subnets.
tenant-connectivity-l3	Used for Layer 3 connectivity changes, including VRFs.
tenant-connectivity-mgmt	Used for tenant in-band and out-of-band management connectivity configurations and for debugging/monitoring policies such as atomic counters and health score.
tenant-epg	Used for managing tenant configurations such as deleting/creating endpoint groups.
tenant-ext-connectivity-l1	Used for write access firmware policies.
tenant-ext-connectivity-l2	Used for managing tenant L2Out configurations.
tenant-ext-connectivity-l3	Used for managing tenant L3Out configurations.
tenant-ext-connectivity-mgmt	Used as write access for firmware policies.
tenant-ext-connectivity-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-ext-protocol-l1	Used for managing tenant external Layer 1 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l2	Used for managing tenant external Layer 2 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l3	Used for managing tenant external Layer 3 protocols such as BGP, OSPF, PIM, and IGMP.
tenant-ext-protocol-mgmt	Used as write access for firmware policies.
tenant-ext-protocol-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-network-profile	Used for managing tenant configurations, such as deleting and creating network profiles, and deleting and creating endpoint groups.
tenant-protocol-l1	Used for managing configurations for Layer 1 protocols under a tenant.
tenant-protocol-l2	Used for managing configurations for Layer 2 protocols under a tenant.
tenant-protocol-l3	Used for managing configurations for Layer 3 protocols under a tenant.
tenant-protocol-mgmt	Only used as write access for firmware policies.
tenant-protocol-ops	Used for tenant traceroute policies.

Role: read-all	
Privilege	**Description**
tenant-QoS	Used for QoS-related configurations for a tenant.
tenant-security	Used for contract-related configurations for a tenant.
vmm-connectivity	Used to read all the objects in Cisco APIC's VMM inventory required for virtual machine connectivity.
vmm-ep	Used to read virtual machine and hypervisor endpoints in the Cisco APIC's VMM inventory.
vmm-policy	Used for managing policies for virtual machine networking.
vmm-protocol-ops	Not used by VMM policies.
vmm-security	Used for managing authentication policies for VMM, such as the username and password for VMware vCenter.

Role: tenant-admin	
Privilege	**Description**
aaa	Used for configuring authentication, authorization, accouting and import/export policies.
access-connectivity-l1	Used for Layer 1 configuration under infra. Example: selectors and port Layer 1 policy configurations.
access-connectivity-l2	Used for Layer 2 configuration under infra. Example: Encap configurations on selectors, and attachable entity.
access-connectivity-l3	Used for Layer 3 configuration under infra and static route configurations under a tenant's L3Out.
access-connectivity-mgmt	Used for management infra policies.
access-connectivity-util	Used for tenant ERSPAN policies.
access-equipment	Used for access port configuration.
access-protocol-l1	Used for Layer 1 protocol configurations under infra.
access-protocol-l2	Used for Layer 2 protocol configurations under infra.
access-protocol-l3	Used for Layer 3 protocol configurations under infra.
access-protocol-mgmt	Used for fabric-wide policies for NTP, SNMP, DNS, and image management.
access-protocol-ops	Used for operations-related access policies such as cluster policy and firmware policies.

Role: tenant-admin	
Privilege	**Description**
access-qos	Used for changing CoPP and QoS-related policies.
fabric-connectivity-l1	Used for Layer 1 configuration under the fabric. Example: selectors and port Layer 1 policy and vPC protection.
fabric-connectivity-l2	Used in firmware and deployment policies for raising warnings for estimating policy deployment impact.
fabric-connectivity-l3	Used for Layer 3 configuration under the fabric. Example: Fabric IPv4, IPv6, and MAC protection groups.
fabric-connectivity-mgmt	Used for atomic counter and diagnostic policies on leaf switches and spine switches.
fabric-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
fabric-equipment	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
fabric-protocol-l1	Used for Layer 1 protocol configurations under the fabric.
fabric-protocol-l2	Used for Layer 2 protocol configurations under the fabric.
fabric-protocol-l3	Used for Layer 3 protocol configurations under the fabric.
fabric-protocol-mgmt	Used for fabric-wide policies for NTP, SNMP, DNS, and image management.
fabric-protocol-ops	Used for ERSPAN and health score policies.
fabric-protocol-util	Used for firmware management traceroute and endpoint tracking policies.
nw-svc-device	Used for managing Layer 4 to Layer 7 service devices.
nw-svc-devshare	Used for managing shared Layer 4 to Layer 7 service devices.
nw-svc-params	Used for managing Layer 4 to Layer 7 service policies.
nw-svc-policy	Used for managing Layer 4 to Layer 7 network service orchestration.
ops	Used for viewing the policies configured including troubleshooting policies. **Note** The **ops** role cannot be used for creating new monitoring and troubleshooting policies. Those policies need to be created by using the **admin** privilege, just like any other configurations in the Cisco APIC.

Role: tenant-admin	
Privilege	**Description**
tenant-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
tenant-connectivity-l2	Used for Layer 2 connectivity changes, including bridge domains and subnets.
tenant-connectivity-l3	Used for Layer 3 connectivity changes, including VRFs.
tenant-connectivity-mgmt	Used for tenant in-band and out-of-band management connectivity configurations and for debugging/monitoring policies such as atomic counters and health score.
tenant-epg	Used for managing tenant configurations such as deleting/creating endpoint groups, VRFs, and bridge domains.
tenant-ext-connectivity-l1	Used for write access firmware policies.
tenant-ext-connectivity-l2	Used for managing tenant L2Out configurations.
tenant-ext-connectivity-l3	Used for managing tenant L3Out configurations.
tenant-ext-connectivity-mgmt	Used as write access for firmware policies.
tenant-ext-connectivity-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-ext-protocol-l1	Used for managing tenant external Layer 1 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l2	Used for managing tenant external Layer 2 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l3	Used for managing tenant external Layer 3 protocols such as BGP, OSPF, PIM, and IGMP.
tenant-ext-protocol-mgmt	Used as Write access for firmware policies.
tenant-ext-protocol-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-network-profile	Used for managing tenant configurations, such as deleting and creating network profiles, and deleting and creating endpoint groups.
tenant-protocol-l1	Used for managing configurations for Layer 1 protocols under a tenant.
tenant-protocol-l2	Used for managing configurations for Layer 2 protocols under a tenant.
tenant-protocol-l3	Used for managing configurations for Layer 3 protocols under a tenant.
tenant-protocol-mgmt	Only used as write access for firmware policies.
tenant-protocol-ops	Used for tenant traceroute policies.

Role: tenant-admin	
Privilege	**Description**
tenant-QoS	Used for QoS-related configurations for a tenant.
tenant-security	Used for contract-related configurations for a tenant.
vmm-connectivity	Used to read all the objects in Cisco APIC's VMM inventory required for virtual machine connectivity.
vmm-ep	Used to read virtual machine and hypervisor endpoints in the Cisco APIC's VMM inventory.
vmm-policy	Used for managing policies for virtual machine networking.
vmm-protocol-ops	Not used by VMM policies.
vmm-security	Used for managing authentication policies for VMM, such as the username and password for VMware vCenter.

Role: tenant-ext-admin	
Privilege	**Description**
tenant-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
tenant-connectivity-l2	Used for Layer 2 connectivity changes, including bridge domains and subnets.
tenant-connectivity-l3	Used for Layer 3 connectivity changes, including VRFs.
tenant-connectivity-mgmt	Used for tenant in-band and out-of-band management connectivity configurations and for debugging/monitoring policies such as atomic counters and health score.
tenant-epg	Used for managing tenant configurations such as deleting/creating endpoint groups, VRFs, and bridge domains.
tenant-ext-connectivity-l1	Used for write access firmware policies.
tenant-ext-connectivity-l2	Used for managing tenant L2Out configurations.
tenant-ext-connectivity-l3	Used for managing tenant L3Out configurations.
tenant-ext-connectivity-mgmt	Used as write access for firmware policies.
tenant-ext-connectivity-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-ext-protocol-l1	Used for managing tenant external Layer 1 protocols. Generally only used for write access for firmware policies.

Role: tenant-ext-admin	
Privilege	**Description**
tenant-ext-protocol-l2	Used for managing tenant external Layer 2 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l3	Used for managing tenant external Layer 3 protocols such as BGP, OSPF, PIM, and IGMP.
tenant-ext-protocol-mgmt	Used as Write access for firmware policies.
tenant-ext-protocol-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-network-profile	Used for managing tenant configurations, such as deleting and creating network profiles, and deleting and creating endpoint groups.
tenant-protocol-l1	Used for managing configurations for Layer 1 protocols under a tenant.
tenant-protocol-l2	Used for managing configurations for Layer 2 protocols under a tenant.
tenant-protocol-l3	Used for managing configurations for Layer 3 protocols under a tenant.
tenant-protocol-mgmt	Only used as write access for firmware policies.
tenant-protocol-ops	Used for tenant traceroute policies.
tenant-QoS	Used for QoS-related configurations for a tenant.
tenant-security	Used for contract-related configurations for a tenant.
vmm-connectivity	Used to read all the objects in Cisco APIC's VMM inventory required for virtual machine connectivity.
vmm-ep	Used to read virtual machine and hypervisor endpoints in the Cisco APIC's VMM inventory.
vmm-policy	Used for managing policies for virtual machine networking.
vmm-protocol-ops	Not used by VMM policies.
vmm-security	Used for managing authentication policies for VMM, such as the username and password for VMware vCenter.

Role: vmm-admin	
Privilege	**Description**
vmm-connectivity	Used to read all the objects in Cisco APIC's VMM inventory required for virtual machine connectivity.
vmm-ep	Used to read virtual machine and hypervisor endpoints in the Cisco APIC's VMM inventory.

Role: vmm-admin	
Privilege	**Description**
vmm-policy	Used for managing policies for virtual machine networking.
vmm-protocol-ops	Not used by VMM policies.
vmm-security	Used for managing authentication policies for a VMM, such as the username and password for VMware vCenter.

Custom Roles

You can create custom roles and assign privileges to the roles. The interface internally assigns one or more privileges to all managed object classes. In an XML model, privileges are assigned in an access attribute. Privilege bits are assigned at compile time and apply per class, and not per instance or object of the class.

In addition to the 45 privilege bits, the "aaa" privilege bit applies to all AAA-subsystem configuration and read operations. The following table provides a matrix of the supported privilege combinations. The rows in the table represent Cisco Application Centric Infrastructure (ACI) modules and the columns represent functionality for a given module. A value of "Yes" in a cell indicates that the functionality for the module is accessible and there exists a privilege bit to access that functionality. An empty cell indicates that the particular functionality for module is not accessible by any privilege bit. See the privilege bit descriptions to learn what each bit does.

	Connectivity	QoS	Security	Application	Fault	Stats	Provider	Service Profile	Service Chain
VMM	Yes		Yes		Yes	Yes	Yes		
Fabric	Yes	Yes	Yes	Yes	Yes	Yes	Yes		
External	Yes	Yes	Yes		Yes	Yes			Yes
Tenant	Yes	Yes	Yes	EPG, NP	Yes	Yes			Yes
Infra	Yes	Yes	Yes	Yes	Yes	Yes			Yes
Ops					Yes	Yes			
Storage	Yes	Yes	Yes	Yes	Yes	Yes			
Network Service	Yes	Yes	Yes	Yes	Yes	Yes		Yes	

Sample RBAC Rules

The RBAC rules in the sample JSON file below enable both trans-tenant access and tenant access to a VMM domain resource. The resources needed by the consumer are `uni/tn-prov1/brc-webCtrct` and `vmmp-Vmware/dom-Datacenter`.

The following two RBAC rules enable the consumer tenant to post the consumer postman query in the JSON file below.

```
<aaaRbacEp>
  <aaaRbacRule objectDn="uni/vmmp-VMware/dom-Datacenter" domain="cons1"/>
  <aaaRbacRule objectDn="uni/tn-prov1/brc-webCtrct" domain="cons1"/>
</aaaRbacEp>
```

The JSON file below contains these two RBAC rules:

```
{"id":"ac62a200-9210-f53b-7114-a8f4cffb9a36","name":"SharedContracts","timestamp":1398806919868,"requests":
[{"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36","id":"2dfc75cc-431e-e136-622c-a577ce7622d8",
"name":"login as prov1",
"description":"",
"url":"http://http://solar.local:8000/api/aaaLogin.json",
"method":"POST",
"headers":"",
"data":
"{\"aaaUser\":{\"attributes\":{\"name\": \"prov1\", \"pwd\": \"secret!\"}}}",
"dataMode":"raw","timestamp":0,"version":2,"time":1398807562828},

{"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36","id":"56e46db0-77ea-743f-a64e-c5f7b1f59807",
"name":"Root login",
"description":"",
"url":"http://http://solar.local:8000/api/aaaLogin.json",
"method":"POST",
"headers":"",
"data":
"{\"aaaUser\":{\"attributes\":{\"name\": \"admin\", \"pwd\": \"secret!\"}}}",
"dataMode":"raw","timestamp":0,"responses":[],"version":2},

{"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36","id":"804893f1-0915-6d35-169d-3af0eb3e64ec",
"name":"consumer tenant only",
"description":"",
"url":"http://http://solar.local:8000/api/policymgr/mo/uni/tn-cons1.xml",
"method":"POST",
"headers":"",
"data":
"<fvTenant name=\"cons1\">
     <aaaDomainRef name=\"cons1\"/>\n
</fvTenant>\n",
"dataMode":"raw","timestamp":0,"version":2,"time":1398968007487},

{"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36","id":"85802d50-8089-bf8b-4481-f149bec258c8",
"name":"login as cons1",
"description":"",
"url":"http://solar.local:8000/api/aaaLogin.json",
"method":"POST",
"headers":"",
"data":
"{\"aaaUser\":{\"attributes\":{\"name\": \"cons1\", \"pwd\": \"secret!\"}}}",
"dataMode":"raw","timestamp":0,"version":2,"time":1398807575531},

{"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36","id":"a2739d92-5f9d-f16c-8894-0f64b6f967a3",
"name":"consumer",
"description":"",
"url":"http://solar.local:8000/api/policymgr/mo/uni/tn-cons1.xml",
"method":"POST","headers":"","data":
"<fvTenant name=\"cons1\" status=\"modified\">\n
    <fvCtx name=\"cons1\"/>\n
     <!-- bridge domain -->\n
        <fvBD name=\"cons1\">\n
  <fvRsCtx tnFvCtxName=\"cons1\" />\n
      <fvSubnet ip=\"10.0.2.128/24\" scope='shared'/>\n
    </fvBD>\n
  \n <!-- DNS Shared Service Contract Interface-->\n
  <vzCPIf name=\"consIf\">\n
```

```
          <vzRsIf tDn=\"uni/tn-prov1/brc-webCtrct\" >\n
    </vzRsIf>\n
  </vzCPIf>\n \n
 <fvAp name=\"cons1\">\n
     <fvAEPg name=\"APP\">\n
          <fvRsBd tnFvBDName=\"cons1\" />\n
   <fvRsNodeAtt tDn=\"topology/pod-1/node-101\" encap=\"vlan-4000\" instrImedcy=\"immediate\"
 mode=\"regular\"/>\n
    <fvRsDomAtt tDn=\"uni/vmmp-VMware/dom-Datacenter\"/>\n
    <fvRsConsIf tnVzCPIfName=\"consIf\"/>\n
  </fvAEPg>\n
    </fvAp>\n
</fvTenant>\n",
"dataMode":"raw","timestamp":0,"version":2,"time":1398818639692},

{"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36","id":"c0bd866d-600a-4f45-46ec-6986398cbf78",
"name":"provider tenant only",
"description":"",
"url":"http://solar.local:8000/api/policymgr/mo/uni/tn-prov1.xml",
"method":"POST",
"headers":"",
"data":
"<fvTenant name=\"prov1\"><aaaDomainRef name=\"prov1\"/>
   \n
</fvTenant>\n",
"dataMode":"raw","timestamp":0,"version":2,"time":1398818137518},

{"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36","id":"d433a213-e95d-646d-895e-3a9e2e2b7ba3",
"name":"create RbacRule",
"description":"",
"url":"http://solar.local:8000/api/policymgr/mo/uni.xml",
"method":"POST",
"headers":"",
"data":
"<aaaRbacEp>\n
     <aaaRbacRule objectDn=\"uni/vmmp-VMware/dom-Datacenter\" domain=\"cons1\"/>\n
     <aaaRbacRule objectDn=\"uni/tn-prov1/brc-webCtrct\" domain=\"cons1\"/>\n
</aaaRbacEp>\n",
"dataMode":"raw","timestamp":0,"version":2,"time":1414195420515},

{"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36","id":"d5c5d580-a11a-7c61-34ac-cbdac249157f",
"name":"provider",
"description":"",
"url":"http://solar.local:8000/api/policymgr/mo/uni/tn-prov1.xml",
"method":"POST",
"headers":"",
"data":
"<fvTenant name=\"prov1\" status=\"modified\">\n
     <fvCtx name=\"prov1\"/>\n
  \n <!-- bridge domain -->\n
      <fvBD name=\"prov1\">\n
       <fvRsCtx tnFvCtxName=\"prov1\" />\n
   </fvBD>\n \n
   <vzFilter name='t0f0' >\n
       <vzEntry etherT='ip' dToPort='10' prot='6' name='t0f0e9' dFromPort='10'>
    </vzEntry>\n
   </vzFilter>\n \n
   <vzFilter name='t0f1'>\n
       <vzEntry etherT='ip' dToPort='209' prot='6' name='t0f1e8' dFromPort='109'>
    </vzEntry>\n
  </vzFilter>\n \n
    <vzBrCP name=\"webCtrct\" scope=\"global\">\n
     <vzSubj name=\"app\">\n
      <vzRsSubjFiltAtt tnVzFilterName=\"t0f0\"/>\n
```

```
   <vzRsSubjFiltAtt tnVzFilterName=\"t0f1\"/>\n
        </vzSubj>\n
    </vzBrCP>\n \n
 <fvAp name=\"prov1AP\">\n
     <fvAEPg name=\"Web\">\n
      <fvRsBd tnFvBDName=\"prov1\" />\n
            <fvRsNodeAtt tDn=\"topology/pod-1/node-17\" encap=\"vlan-4000\"
instrImedcy=\"immediate\" mode=\"regular\"/>\n
            <fvRsProv tnVzBrCPName=\"webCtrct\"/>\n
   <fvRsDomAtt tDn=\"uni/vmmp-VMware/dom-Datacenter\"/>\n
            <fvSubnet ip=\"10.0.1.128/24\" scope='shared'/>\n
  </fvAEPg>\n
 </fvAp>\n
</fvTenant>\n",
"dataMode":"raw","timestamp":0,"version":2,"time":1398818660457},

{"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36","id":"e8866493-2188-8893-8e0c-4ca0903b18b8",
"name":"add user prov1",
"description":"",
"url":"http://solar.local:8000/api/policymgr/mo/uni/userext.xml",
"method":"POST",
"headers":"",
"data":
"<aaaUserEp>\n
     <aaaUser name=\"prov1\" pwd=\"secret!\">
      <aaaUserDomain name=\"prov1\">
          <aaaUserRole name=\"tenant-admin\" privType=\"writePriv\"/>
          <aaaUserRole name=\"vmm-admin\" privType=\"writePriv\"/>
  </aaaUserDomain>
  </aaaUser>\n
      <aaaUser name=\"cons1\" pwd=\"secret!\">
   <aaaUserDomain name=\"cons1\">
   <aaaUserRole name=\"tenant-admin\" privType=\"writePriv\"/>
   <aaaUserRole name=\"vmm-admin\" privType=\"writePriv\"/>
   </aaaUserDomain>
  </aaaUser>\n
      <aaaDomain name=\"prov1\"/>\n
   <aaaDomain name=\"cons1\"/>\n
</aaaUserEp>\n",
"dataMode":"raw","timestamp":0,"version":2,"time":1398820966635}]}
```


Enabling Port Security

About Port Security and ACI

The port security feature protects the ACI fabric from being flooded with unknown MAC addresses by limiting the number of MAC addresses learned per port. The port security feature support is available for physical ports, port channels, and virtual port channels.

Port Security Guidelines and Restrictions

The guidelines and restrictions are as follows:

- Port security is available per port.
- Port security is supported for physical ports, port channels, and virtual port channels (vPCs).

- Static and dynamic MAC addresses are supported.
- MAC address moves are supported from secured to unsecured ports and from unsecured ports to secured ports.
- The MAC address limit is enforced only on the MAC address and is not enforced on a MAC and IP address.
- Port security is not supported with the Fabric Extender (FEX).

Port Security and Learning Behavior

For non-vPC ports or port channels, whenever a learn event comes for a new endpoint, a verification is made to see if a new learn is allowed. If the corresponding interface has a port security policy not configured or disabled, the endpoint learning behavior is unchanged with what is supported. If the policy is enabled and the limit is reached, the current supported action is as follows:

- Learn the endpoint and install it in the hardware with a drop action.
- Silently discard the learn.

If the limit is not reached, the endpoint is learned and a verification is made to see if the limit is reached because of this new endpoint. If the limit is reached, and the learn disable action is configured, learning will be disabled in the hardware on that interface (on the physical interface or on a port channel or vPC). If the limit is reached and the learn disable action is not configured, the endpoint will be installed in hardware with a drop action. Such endpoints are aged normally like any other endpoints.

When the limit is reached for the first time, the operational state of the port security policy object is updated to reflect it. A static rule is defined to raise a fault so that the user is alerted. A syslog is also raised when the limit is reached.

In case of vPC, when the MAC limit is reached, the peer leaf switch is also notified so learning can be disabled on the peer. As the vPC peer can be rebooted any time or vPC legs can become unoperational or restart, this state will be reconciled with the peer so vPC peers do not go out of sync with this state. If they get out of sync, there can be a situation where learning is enabled on one leg and disabled on the other leg.

By default, once the limit is reached and learning is disabled, it will be automatically re-enabled after the default timeout value of 60 seconds.

Port Security at Port Level

In the APIC, the user can configure the port security on switch ports. Once the MAC limit has exceeded the maximum configured value on a port, all traffic from the exceeded MAC addresses is forwarded. The following attributes are supported:

- **Port Security Timeout**—The current supported range for the timeout value is from 60 to 3600 seconds.
- **Violation Action**—The violation action is available in protect mode. In the protect mode, MAC learning is disabled and MAC addresses are not added to the CAM table. Mac learning is re-enabled after the configured timeout value.
- **Maximum Endpoints**—The current supported range for the maximum endpoints configured value is from 0 to 12000. If the maximum endpoints value is 0, the port security policy is disabled on that port.

Protect Mode

The protect mode prevents further port security violations from occurring. Once the MAC limit exceeds the maximum configured value on a port, all traffic from excess MAC addresses will be dropped and further learning is disabled.

Configuring Port Security Using REST API

Configure the port security.

Example:

```
<polUni>
 <infraInfra>

    <l2PortSecurityPol name="testL2PortSecurityPol" maximum="10" violation=”protect” timeout=“300"/>


    <infraNodeP name="test">
        <infraLeafS name="test" type="range">
            <infraNodeBlk name="test" from_="101" to_="102"/>
        </infraLeafS>
        <infraRsAccPortP tDn="uni/infra/accportprof-test"/>
    </infraNodeP>

            <infraAccPortP name="test">
        <infraHPortS name="pselc" type="range">
           <infraPortBlk name="blk"
                                fromCard="1" toCard="1" fromPort="20" toPort="22">
                 </infraPortBlk>
            <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-testPortG" />
         </infraHPortS>
            </infraAccPortP>

            <infraFuncP>
            <infraAccPortGrp name="testPortG">
                <infraRsL2PortSecurityPol tnL2PortSecurityPolName="testL2PortSecurityPol"/>
                <infraRsAttEntP tDn="uni/infra/attentp-test" />
            </infraAccPortGrp>
            </infraFuncP>

    <infraAttEntityP name="test">
        <infraRsDomP tDn="uni/phys-mininet"/>
    </infraAttEntityP>
 </infraInfra>
</polUni>
```

Enabling COOP Authentication

Overview

Council of Oracle Protocol (COOP) is used to communicate the mapping information (location and identity) to the spine proxy. A leaf switch forwards endpoint address information to the spine switch 'Oracle' using Zero Message Queue (ZMQ). COOP running on the spine nodes will ensure all spine nodes maintain a consistent copy of endpoint address and location information and additionally maintain the distributed hash table (DHT) repository of endpoint identity to location mapping database.

COOP data path communication provides high priority to transport using secured connections. COOP is enhanced to leverage the MD5 option to protect COOP messages from malicious traffic injection. The APIC controller and switches support COOP protocol authentication.

COOP protocol is enhanced to support two ZMQ authentication modes: strict and compatible.

- Strict mode: COOP allows MD5 authenticated ZMQ connections only.
- Compatible mode: COOP accepts both MD5 authenticated and non-authenticated ZMQ connections for message transportation.

Using COOP with Cisco APIC

To support COOP Zero Message Queue (ZMQ) authentication support across the Cisco Application Centric Infrastructure (ACI) fabric, the Application Policy Infrastructure Controller (APIC) supports the MD5 password and also supports the COOP secure mode.

COOP ZMQ Authentication Type Configuration—A new managed object, `coop:AuthP`, is added to the Data Management Engine (DME)/COOP database (coop/inst/auth). The default value for the attribute type is "compatible", and users have the option to configure the type to be "strict".

COOP ZMQ Authentication MD5 password—The APIC provides a managed object (`fabric:SecurityToken`), that includes an attribute to be used for the MD5 password. An attribute in this managed object, called "token", is a string that changes every hour. COOP obtains the notification from the DME to update the password for ZMQ authentication. The attribute token value is not displayed.

Guidelines and Limitations

Follow these guidelines and limitations:

- During an ACI fabric upgrade, the COOP strict mode is disallowed until all switches are upgraded. This protection prevents the unexpected rejection of a COOP connection that could be triggered by prematurely enabling the strict mode.

Configuring COOP Authentication Using the REST API

Configure a COOP authentication policy.

In the example, the strict mode is chosen.

Example:

```
https://172.23.53.xx/api/node/mo/uni/fabric/pol-default.xml

<coopPol type="strict">
</coopPol>
```

Enabling Control Plane Policing

About Control Plane Policing

Control plane policing (CoPP) protects the control plane, which ensures network stability, reachability, and packet delivery.

This feature allows specification of parameters, for each protocol that can reach the control processor to be rate-limited using a policer. The policing is applied to all traffic destined to any of the IP addresses of the router or Layer 3 switch. A common attack vector for network devices is the denial-of-service (DoS) attack, where excessive traffic is directed at the device interfaces.

The Cisco Application Centric Infrastructure (ACI) leaf and spine switch NX-OS provides CoPP to prevent DoS attacks from impacting performance. Such attacks, which can be perpetrated either inadvertently or maliciously, typically involve high rates of traffic destined to the supervisor module of a Cisco ACI leaf and spine switch CPU or CPU itself.

The supervisor module of Cisco ACI leaf and spine switch switches divides the traffic that it manages into two functional components or planes:

- **Data plane**—Handles all the data traffic. The basic functionality of a Cisco NX-OS device is to forward packets from one interface to another. The packets that are not meant for the switch itself are called the transit packets. These packets are handled by the data plane.
- **Control plane**—Handles all routing protocol control traffic. These protocols, such as the Border Gateway Protocol (BGP) and the Open Shortest Path First (OSPF) Protocol, send control packets between devices. These packets are destined to router addresses and are called control plane packets.

The Cisco ACI leaf and spine switch supervisor module has a control plane and is critical to the operation of the network. Any disruption or attacks to the supervisor module will result in serious network outages. For example, excessive traffic to the supervisor module could overload and slow down the performance of the entire Cisco ACI fabric. Another example is a DoS attack on the Cisco ACI leaf and spine switch supervisor module that could generate IP traffic streams to the control plane at a very high rate, forcing the control plane to spend a large amount of time in handling these packets and preventing the control plane from processing genuine traffic.

Examples of DoS attacks are as follows:

- Internet Control Message Protocol (ICMP) echo requests
- IP fragments
- TCP SYN flooding

These attacks can impact the device performance and have the following negative effects:

- Reduced service quality (such as poor voice, video, or critical applications traffic)
- High route processor or switch processor CPU utilization
- Route flaps due to loss of routing protocol updates or keepalives
- Processor resource exhaustion, such as the memory and buffers
- Indiscriminate drops of incoming packets

Note Cisco ACI leaf and spine switches are by default protected by CoPP with default settings. This feature allows for tuning the parameters on a group of nodes based on customer needs.

Control Plane Protection

To protect the control plane, the Cisco NX-OS running on Cisco ACI leaf and spine switches segregates different packets destined for the control plane into different classes. Once these classes are identified, the Cisco NX-OS device polices the packets, which ensures that the supervisor module is not overwhelmed.

Control Plane Packet Types:

Different types of packets can reach the control plane:

- **Receive Packets**—Packets that have the destination address of a router. The destination address can be a Layer 2 address (such as a router MAC address) or a Layer 3 address (such as the IP address of a router interface). These packets include router updates and keepalive messages. Multicast packets can also be in this category where packets are sent to multicast addresses that are used by a router.
- **Exception Packets**—Packets that need special handling by the supervisor module. For example, if a destination address is not present in the Forwarding Information Base (FIB) and results in a miss, the supervisor module sends an ICMP unreachable packet back to the sender. Another example is a packet with IP options set.
- **Redirect Packets**—Packets that are redirected to the supervisor module. Features such as Dynamic Host Configuration Protocol (DHCP) snooping or dynamic Address Resolution Protocol (ARP) inspection redirect some packets to the supervisor module.
- **Glean Packets**—If a Layer 2 MAC address for a destination IP address is not present in the FIB, the supervisor module receives the packet and sends an ARP request to the host.

All of these different packets could be maliciously used to attack the control plane and overwhelm the Cisco ACI fabric. CoPP classifies these packets to different classes and provides a mechanism to individually control the rate at which the Cisco ACI leaf and spine switch supervisor module receives these packets.

Classification for CoPP:

For effective protection, the Cisco ACI leaf and spine switch NX-OS classifies the packets that reach the supervisor modules to allow you to apply different rate controlling policies based on the type of the packet. For example, you might want to be less strict with a protocol packet such as Hello messages but more strict with a packet that is sent to the supervisor module because the IP option is set.

Available Protocols:

Protocol	Description
Glean	With this protocol, when the bridge domain is in proxy mode, unknown unicast traffic received by the leaf switch is sent to the hardware proxy (the spine switch). The spine switch changes the eth-type of the packet to a special eth-type (0xfff2). When these packets reach the leaf switches through the fabric ports, the packets are classified under glean. The packets are sent to the leaf switch's CPU, and the leaf switch's CPU generates an ARP request for the connected external devices.
ToR Glean	ToR glean activates when an endpoint moves or is cleared because of link flap and does not update the source leaf switch's remote IP address endpoint entry. A packet egresses the source leaf switch with the destination leaf switch's TEP address. On the destination leaf switch, because of the missing local IP address entry, the packet gets sent to the leaf switch CPU to generate an ARP request for those IP addresses. These packets are classified under ToR glean.

Rate Controlling Mechanisms:

Once the packets are classified, the Cisco ACI leaf and spine switch NX-OS has different mechanisms to control the rate at which packets arrive at the supervisor module.

You can configure the following parameters for policing:

- **Committed information rate (CIR)**—Desired bandwidth, specified as a bit rate or a percentage of the link rate.
- **Committed burst (BC)**—Size of a traffic burst that can exceed the CIR within a given unit of time and not impact scheduling.

Default Policing Policies:

When the Cisco ACI leaf and spine switch is bootup, the platform setup pre-defined CoPP parameters for different protocols are based on the tests done by Cisco.

Guidelines and Limitations for CoPP

CoPP has the following configuration guidelines and limitations:

- We recommend that you use the default CoPP policy initially and then later modify the CoPP policies based on the data center and application requirements.
- Customizing CoPP is an ongoing process. CoPP must be configured according to the protocols and features used in your specific environment as well as the supervisor features that are required by the server environment. As these protocols and features change, CoPP must be modified.

- We recommend that you continuously monitor CoPP. If drops occur, determine if CoPP dropped traffic unintentionally or in response to a malfunction or attack. In either event, analyze the situation and evaluate the need to modify the CoPP policies.
- You must ensure that the CoPP policy does not filter critical traffic such as routing protocols or interactive access to the device. Filtering this traffic could prevent remote access to the Cisco ACI Leaf/Spine and require a console connection.
- Do not mis-configure CoPP pre-filter entries. CoPP pre-filter entries might impact connectivity to multi-pod configurations, remote leaf switches, and Cisco ACI Multi-Site deployments.
- You can use the APIC UI to be able to tune the CoPP parameters.
- Per interface per protocol is only supported on Leaf switches.
- FEX ports are not supported on per interface per protocol.
- For per interface per protocol the supported protocols are; ARP, ICMP, CDP, LLDP, LACP, BGP, STP, BFD, and OSPF.
- The TCAM entry maximum for per interface per protocol is 256. Once the threshold is exceeded a fault will be raised.

Configuring CoPP Using the REST API

Step 1 Configure a CoPP leaf profile:

Example:

```
<!-- api/node/mo/uni/.xml -->
<infraInfra>
  <coppLeafProfile type="custom" name="mycustom">                <!-- define copp leaf profile -->

    <coppLeafGen1CustomValues bgpBurst="150" bgpRate="300"/>
  </coppLeafProfile>
  <infraNodeP name="leafCopp">
    <infraLeafS name="leafs" type="range">
      <infraNodeBlk name="leaf1" from_="101" to_="101"/>
      <infraNodeBlk name="leaf3" from_="103" to_="103"/>
      <infraRsAccNodePGrp tDn="uni/infra/funcprof/accnodepgrp-myLeafCopp"/>
    </infraLeafS>
  </infraNodeP>
  <infraFuncP>
    <infraAccNodePGrp name="myLeafCopp">
     <infraRsLeafCoppProfile tnCoppLeafProfileName="mycustom"/>   <!-- bind copp leaf policy to leaf
 </infraAccNodePGrp>                                                 profile -->
  </infraFuncP>
</infraInfra>
```

Step 2 Configure a CoPP spine profile:

Example:

```
<!-- api/node/mo/uni/.xml -->
<infraInfra>
  <coppSpineProfile type="custom" name="mycustomSpine">            <!-- define copp leaf profile
-->
    <coppSpineGen1CustomValues bgpBurst="150" bgpRate="300"/>
  </coppSpineProfile>
```

```
  <infraSpineP name="spineCopp">
    <infraSpineS name="spines" type="range">
      <infraNodeBlk name="spine1" from_="104" to_="104"/>
      <infraRsSpineAccNodePGrp tDn="uni/infra/funcprof/spaccnodepgrp-mySpineCopp"/>
    </infraSpineS>
  </infraSpineP>
  <infraFuncP>
    <infraSpineAccNodePGrp name="mySpineCopp">
      <infraRsSpineCoppProfile tnCoppSpineProfileName="mycustomSpine"/> <!-- bind copp spine policy
 to
    </infraSpineAccNodePGrp>                                                    spine profile -->
  </infraFuncP>
</infraInfra>
```

Configuring CoPP Per Interface Per Protocol Using REST API

Configure a CoPP per interface per protocol:

Example:

```
<polUni>
    <infraInfra>
    <infraNodeP name="default">
        <infraLeafS name="default" type="range">
            <infraNodeBlk name="default" to_="101" from_="101"/>
        </infraLeafS>
        <infraRsAccPortP tDn="uni/infra/accportprof-default"/>
    </infraNodeP>
    <infraAccPortP name="default">
        <infraHPortS name="regularPorts" type="range">
            <infraPortBlk name="blk1" toPort="7" fromPort="1" toCard="1" fromCard="1"/>
                <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-copp"/>
        </infraHPortS>
    </infraAccPortP>

    <infraFuncP>
        <infraAccPortGrp name="copp">
            <infraRsCoppIfPol tnCoppIfPolName="pc"/>
        </infraAccPortGrp>
    </infraFuncP>

    <coppIfPol name = "pc" >
        <coppProtoClassP name = "test" matchProto="lldp,arp" rate="505" burst = "201"/>
        <coppProtoClassP name = "test1" matchProto="bgp" rate="500" burst = "200" />
    </coppIfPol>
</infraInfra>
</polUni>
```

Configuring First Hop Security

About First Hop Security

First-Hop Security (FHS) features enable a better IPv4 and IPv6 link security and management over the layer 2 links. In a service provider environment, these features closely control address assignment and derived operations, such as Duplicate Address Detection (DAD) and Address Resolution (AR).

The following supported FHS features secure the protocols and help build a secure endpoint database on the fabric leaf switches, that are used to mitigate security threats such as MIM attacks and IP thefts:

- ARP Inspection—allows a network administrator to intercept, log, and discard ARP packets with invalid MAC address to IP address bindings.
- ND Inspection—learns and secures bindings for stateless autoconfiguration addresses in Layer 2 neighbor tables.
- DHCP Inspection—validates DHCP messages received from untrusted sources and filters out invalid messages.
- RA Guard—allows the network administrator to block or reject unwanted or rogue router advertisement (RA) guard messages.
- IPv4 and IPv6 Source Guard—blocks any data traffic from an unknown source.
- Trust Control—a trusted source is a device that is under your administrative control. These devices include the switches, routers, and servers in the Fabric. Any device beyond the firewall or outside the network is an untrusted source. Generally, host ports are treated as untrusted sources.

FHS features provide the following security measures:

- **Role Enforcement**—Prevents untrusted hosts from sending messages that are out the scope of their role.
- **Binding Enforcement**—Prevents address theft.
- **DoS Attack Mitigations**—Prevents malicious end-points to grow the end-point database to the point where the database could stop providing operation services.
- **Proxy Services**—Provides some proxy-services to increase the efficiency of address resolution.

FHS features are enabled on a per tenant bridge domain (BD) basis. As the bridge domain, may be deployed on a single or across multiple leaf switches, the FHS threat control and mitigation mechanisms cater to a single switch and multiple switch scenarios.

ACI FHS Deployment

Most FHS features are configured in a two-step fashion: firstly you define a policy which describes the behavior of the feature, secondly you apply this policy to a "domain" (being the Tenant Bridge Domain or the Tenant Endpoint Group). Different policies that define different behaviors can be applied to different intersecting domains. The decision to use a specific policy is taken by the most specific domain to which the policy is applied.

The policy options can be defined from the Cisco APIC GUI found under the Tenant_*name*>Networking>Protocol Policies>First Hop Security tab.

Guidelines and Limitations

Follow these guidelines and limitations:

- Starting with release 3.1(1), FHS is supported with virtual Endpoints (AVS only).
- FHS feature is not supported when an EPG is deployed with VXLAN encapsulation.
- Any secured endpoint entry in the FHS Binding Table Database in **DOWN** state will get cleared after **18 Hours** of timeout. The entry moves to **DOWN** state when the front panel port where the entry is learned is link down. During this window of **18 Hours**, if the endpoint is moved to a different location and is seen on a different port, the entry will be gracefully moved out of **DOWN** state to **REACHABLE/STALE** as long as the endpoint is reachable from the other port it is moved from.
- When IP Source Guard is enabled, the IPv6 traffic that is sourced using IPv6 Link Local address as IP source address is not subject to the IP Source Guard enforcement (i.e. Enforcement of Source Mac <=> Source IP Bindings secured by IP Inspect Feature). This traffic is permitted by default irrespective of binding check failures.
- FHS is not supported on L3Out interfaces.
- FHS is not supported N9K-M12PQ based TORs.
- FHS in ACI Multi-Site is a site local capability therefore it can only be enabled in a site from the APIC cluster. Also, FHS in ACI Multi-Site only works when the BD and EPG is site local and not stretched across sites. FHS security cannot be enabled for stretched BD or EPGs.
- FHS is not supported on a Layer 2 only bridge domain.
- Enabling FHS feature can disrupt traffic for 50 seconds because the EP in the BD are flushed and EP Learning in the BD is disabled for 50 seconds.

Configuring FHS in APIC Using REST API

Before you begin

- The tenant and bridge domain must be configured.

Configure the FHS and Trust Control policies.

Example:

```
<polUni>
    <fvTenant name="Coke">
        <fhsBDPol name="bdpol5" ipInspectAdminSt="enabled-ipv6" srcGuardAdminSt="enabled-both"
raGuardAdminSt="enabled" status="">
            <fhsRaGuardPol name="raguard5" managedConfigCheck="true" managedConfigFlag="true"
otherConfigCheck="true" otherConfigFlag="true" maxRouterPref="medium" minHopLimit="3" maxHopLimit="15"
 status=""/>
        </fhsBDPol>
        <fvBD name="bd3">
            <fvRsBDToFhs tnFhsBDPolName="bdpol5" status=""/>
```

```
        </fvBD>
    </fvTenant>
</polUni>

<polUni>
<fvTenant name="Coke">
        <fhsTrustCtrlPol name="trustctrl5" hasDhcpv4Server="true" hasDhcpv6Server="true"
hasIpv6Router="true" trustRa="true" trustArp="true" trustNd="true" />
        <fvAp name="wwwCokecom3">
           <fvAEPg name="test966">
           <fvRsTrustCtrl tnFhsTrustCtrlPolName="trustctrl5" status=""/>
        </fvAEPg>
        </fvAp>
</fvTenant>
</polUni>
```

Configuring 802.1x

802.1X Overview

802.1X defines a client-server based access control and authentication protocol that restricts unauthorized clients from connecting to a LAN through publicly accessible ports. The authentication server authenticates each client connected to a Cisco NX-OS device port.

Until the client is authenticated, 802.1X access control allows only Extensible Authentication Protocol over LAN (EAPOL) traffic through the port to which the client is connected. After authentication is successful, normal traffic can pass through the port.

The RADIUS distributed client/server system allows you to secure networks against unauthorized access. In the Cisco ACI implementation, RADIUS clients run on the ToRs and send authentication and accounting requests to a central RADIUS server that contains all user authentication and network service access information.

Host Support

The 802.1X feature can restrict traffic on a port with the following modes:

- **Single-host Mode**—Allows traffic from only one endpoint device on the 802.1X port. Once the endpoint device is authenticated, the APIC puts the port in the authorized state. When the endpoint device leaves the port, the APIC put the port back into the unauthorized state. A security violation in 802.1X is defined as a detection of frames sourced from any MAC address other than the single MAC address authorized as a result of successful authentication. In this case, the interface on which this security association violation is detected (EAPOL frame from the other MAC address) will be disabled. Single host mode is applicable only for host-to-switch topology and when a single host is connected to the Layer 2 (Ethernet access port) or Layer 3 port (routed port) of the APIC.
- **Multi-host Mode**—Allows multiple hosts per port but only the first one gets authenticated. The port is moved to the authorized state after the successful authorization of the first host. Subsequent hosts are not required to be authorized to gain network access once the port is in the authorized state. If the port becomes unauthorized when reauthentication fails or an EAPOL logoff message is received, all attached hosts are denied access to the network. The capability of the interface to shut down upon security

association violation is disabled in multiple host mode. This mode is applicable for both switch-to-switch and host-to-switch topologies

- **Multi-Auth Mode**—Allows multiple hosts and all hosts are authenticated separately.

Note Each host must have the same EPG/VLAN information.

- **Multi-Domain Mode**—For separate data and voice domain. For use with IP-Phones.

Authentication Modes

ACI 802.1X supports the following authentication modes:

- **EAP**—The authenticator then sends an EAP-request/identity frame to the supplicant to request its identity (typically, the authenticator sends an initial identity/request frame followed by one or more requests for authentication information). When the supplicant receives the frame, it responds with an EAP-response/identity frame.
- **MAB**—MAC Authentication Bypass (MAB) is supported as the fallback authentication mode. MAB enables port-based access control using the MAC address of the endpoint. A MAB-enabled port can be dynamically enabled or disabled based on the MAC address of the device that connects to it. Prior to MAB, the endpoint's identity is unknown and all traffic is blocked. The switch examines a single packet to learn and authenticate the source MAC address. After MAB succeeds, the endpoint's identity is known and all traffic from that endpoint is allowed. The switch performs source MAC address filtering to help ensure that only the MAB-authenticated endpoint is allowed to send traffic.

Guidelines and Limitations

802.1X port-based authentication has the following configuration guidelines and limitations:

- The Cisco ACI supports 802.1X authentication only on physical ports.
- The Cisco ACI does not support 802.1X authentication on port channels or subinterfaces.
- The Cisco ACI supports 802.1X authentication on member ports of a port channel but not on the port channel itself.
- Member ports with and without 802.1X configuration can coexist in a port channel. However, you must ensure the identical 802.1X configuration on all the member ports in order for channeling to operate with 802.1X
- When you enable 802.1X authentication, supplicants are authenticated before any other Layer 2 or Layer 3 features are enabled on an Ethernet interface.
- 802.1X is supported only on a leaf chassis that is EX or FX type.
- 802.1X is only supported Fabric Access Ports. 802.1X is not supported on Port-Channels, or Virtual-Port-Channels.
- IPv6 is not supported for dot1x clients in the 3.2(1) release.

- While downgrading to earlier releases especially in cases where certain interface config (host mode and auth type) is unsupported in that release, dot1x authentication type defaults to none. Host-mode would need to be manually re-configured to either single host/multi host depending on whatever is desired. This is to ensure that the user configures only the supported modes/auth-types in that release and doesn't run into unsupported scenarios.
- Multi-Auth supports 1 voice client and multiple data clients (all belonging to same data vlan/epg).
- Fail-epg/vlan under 802.1X node authentication policy is a mandatory configuration.
- Multi-domain more than 1 voice and 1 data client puts the port in security disabled state.
- The following platforms are not supported for 802.1X:
  - N9K-C9396PX
  - N9K-M12PQ
  - N9K-C93128TX
  - N9K-M12PQ

Configuration Overview

The 802.1X and RADIUS processes are started only when enabled by APIC. Internally, this means dot1x process is started when 802.1X Inst MO is created and radius process is created when radius entity is created. Dot1x based authentication must be enabled on each interface for authenticating users connected on that interface otherwise the behavior is unchanged.

RADIUS server configuration is done separately from dot1x configuration. RADIUS configuration defines a list of RADIUS servers and a way to reach them. Dot1x configuration contains a reference to RADIUS group (or default group) to use for authentication.

Both 802.1X and RADIUS configuration must be done for successful authentication. Order of configuration is not important but if there is no RADIUS configuration then 802.1X authentication cannot be successful.

Configuring 802.1X Node Authentication Using the REST API

Configure a 802.1X node authentication policy:

Example:

```
<polUni>
<infraInfra>
  <l2NodeAuthPol annotation="" descr="" dn="uni/infra/nodeauthpol-802-node-2"
failAuthEpg="tn-t2,ap-ap,epg-epg1" failAuthVlan="vlan-2078" name="802-node-2" nameAlias="" ownerKey=""
 ownerTag="">
<l2RsAaaRadiusProviderGroup annotation="" tDn="uni/userext/radiusext/radiusprovidergroup-radius-grp"/>
</l2NodeAuthPol>
</infraInfra>
</polUni>

Modify:
<polUni>
<infraInfra>
  <l2NodeAuthPol annotation="" descr="" dn="uni/infra/nodeauthpol-802-node-2"
failAuthEpg="tn-t2,ap-ap,epg-epg1" failAuthVlan="vlan-2066" name="802-node-2" nameAlias="" ownerKey=""
```

```
 ownerTag="" status="deleted">
<l2RsAaaRadiusProviderGroup annotation="" tDn="uni/userext/radiusext/radiusprovidergroup-radius-grp"/>
</l2NodeAuthPol>
</infraInfra>
</polUni>

Delete:
<polUni>
<infraInfra>
  <l2NodeAuthPol annotation="" descr="" dn="uni/infra/nodeauthpol-802-node-2"
failAuthEpg="tn-t2,ap-ap,epg-epg1" failAuthVlan="vlan-2078" name="802-node-2" nameAlias="" ownerKey=""
 ownerTag="" status="deleted">
<l2RsAaaRadiusProviderGroup annotation="" tDn="uni/userext/radiusext/radiusprovidergroup-radius-grp"
 status="deleted"/>
</l2NodeAuthPol>
</infraInfra>
</polUni>
```

Configuring 802.1X Port Authentication Using the REST API

Create a 802.1X port authentication policy:

Example:

```
<polUni>
<infraInfra>
  <l2PortAuthPol adminSt="enabled" annotation="" descr="" dn="uni/infra/portauthpol-test21"
hostMode="multi-auth" name="test21" nameAlias="" ownerKey="" ownerTag="">
    <l2PortAuthCfgPol annotation="" macAuth="bypass" maxReauthReq="2" maxReq="2"   reAuthPeriod="3600"
 serverTimeout="30" suppTimeout="30" txPeriod="30"/>
  </l2PortAuthPol>
</infraInfra>
</polUni>

Modify:
<polUni>
<infraInfra>
  <l2PortAuthPol adminSt="enabled" annotation="" descr="" dn="uni/infra/portauthpol-test21"
hostMode="multi-domain" name="test21" nameAlias="" ownerKey="" ownerTag="" >
    <l2PortAuthCfgPol annotation="" macAuth="eap" maxReauthReq="2" maxReq="2"   reAuthPeriod="3600"
 serverTimeout="30" suppTimeout="30" txPeriod="30"/>
  </l2PortAuthPol>
</infraInfra>
</polUni>

Delete:
<polUni>
<infraInfra>
  <l2PortAuthPol adminSt="enabled" annotation="" descr="" dn="uni/infra/portauthpol-test21"
hostMode="multi-host" name="test21" nameAlias="" ownerKey="" ownerTag="" status="deleted">
    <l2PortAuthCfgPol annotation="" macAuth="bypass" maxReauthReq="2" maxReq="2"   reAuthPeriod="3600"
 serverTimeout="30" suppTimeout="30" txPeriod="30" status="deleted"/>
  </l2PortAuthPol>
</infraInfra>
</polUni>
```