Access, Authentication, and Accounting

This chapter contains the following sections:

- Overview, on page 1
- Configuration, on page 22

Overview

User Access, Authorization, and Accounting

Application Policy Infrastructure Controller (APIC) policies manage the authentication, authorization, and accounting (AAA) functions of the Cisco Application Centric Infrastructure (ACI) fabric. The combination of user privileges, roles, and domains with access rights inheritance enables administrators to configure AAA functions at the managed object level in a granular fashion. These configurations can be implemented using the REST API, the CLI, or the GUI.

Multiple Tenant Support

A core Application Policy Infrastructure Controller (APIC) internal data access control system provides multitenant isolation and prevents information privacy from being compromised across tenants. Read/write restrictions prevent any tenant from seeing any other tenant's configuration, statistics, faults, or event data. Unless the administrator assigns permissions to do so, tenants are restricted from reading fabric configuration, policies, statistics, faults, or events.

User Access: Roles, Privileges, and Security Domains

The APIC provides access according to a user’s role through role-based access control (RBAC). An Cisco Application Centric Infrastructure (ACI) fabric user is associated with the following:

- A set of roles
- For each role, a privilege type: no access, read-only, or read-write
- One or more security domain tags that identify the portions of the management information tree (MIT) that a user can access

The ACI fabric manages access privileges at the managed object (MO) level. A privilege is an MO that enables or restricts access to a particular function within the system. For example, fabric-equipment is a privilege bit.
This bit is set by the Application Policy Infrastructure Controller (APIC) on all objects that correspond to equipment in the physical fabric.

A role is a collection of privilege bits. For example, because an “admin” role is configured with privilege bits for “fabric-equipment” and “tenant-security,” the “admin” role has access to all objects that correspond to equipment of the fabric and tenant security.

A security domain is a tag associated with a certain subtree in the ACI MIT object hierarchy. For example, the default tenant “common” has a domain tag common. Similarly, the special domain tag all includes the entire MIT object tree. An administrator can assign custom domain tags to the MIT object hierarchy. For example, an administrator could assign the “solar” domain tag to the tenant named solar. Within the MIT, only certain objects can be tagged as security domains. For example, a tenant can be tagged as a security domain but objects within a tenant cannot.

**Note**  
Security Domain password strength parameters can be configured by creating Custom Conditions or by selecting Any Three Conditions that are provided.

Creating a user and assigning a role to that user does not enable access rights. It is necessary to also assign the user to one or more security domains. By default, the ACI fabric includes two special pre-created domains:

- **All**—allows access to the entire MIT
- **Infra**—allows access to fabric infrastructure objects/subtrees, such as fabric access policies

**Note**  
For read operations to the managed objects that a user's credentials do not allow, a "DN/Class Not Found" error is returned, not "DN/Class Unauthorized to read." For write operations to a managed object that a user's credentials do not allow, an HTTP 401 Unauthorized error is returned. In the GUI, actions that a user's credentials do not allow, either they are not presented, or they are grayed out.

A set of predefined managed object classes can be associated with domains. These classes should not have overlapping containment. Examples of classes that support domain association are as follows:

- **Layer 2 and Layer 3 network managed objects**
- **Network profiles (such as physical, Layer 2, Layer 3, management)**
- **QoS policies**

When an object that can be associated with a domain is created, the user must assign domain(s) to the object within the limits of the user's access rights. Domain assignment can be modified at any time.

If a virtual machine management (VMM) domain is tagged as a security domain, the users contained in the security domain can access the correspondingly tagged VMM domain. For example, if a tenant named solar is tagged with the security domain called sun and a VMM domain is also tagged with the security domain called sun, then users in the solar tenant can access the VMM domain according to their access rights.

**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 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](https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/apicroles/roles.html)

<table>
<thead>
<tr>
<th>Role</th>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa</td>
<td>aaa</td>
<td>Used for configuring authentication, authorization, accounting, and import/export policies.</td>
</tr>
<tr>
<td>admin</td>
<td>admin</td>
<td>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.</td>
</tr>
</tbody>
</table>

#### Role: access-admin

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access-connectivity-l1</td>
<td>Used for Layer 1 configuration under infra. Example: selectors and port Layer 1 policy configurations.</td>
</tr>
<tr>
<td>access-connectivity-l2</td>
<td>Used for Layer 2 configuration under infra. Example: encap configurations on selectors, and attachable entity.</td>
</tr>
<tr>
<td>access-connectivity-l3</td>
<td>Used for Layer 3 configuration under infra and static route configurations under a tenant's L3Out.</td>
</tr>
<tr>
<td>access-connectivity-mgmt</td>
<td>Used for management infra policies.</td>
</tr>
<tr>
<td>access-connectivity-util</td>
<td>Used for tenant ERSPAN policies.</td>
</tr>
<tr>
<td>access-equipment</td>
<td>Used for access port configuration.</td>
</tr>
<tr>
<td>access-protocol-l1</td>
<td>Used for Layer 1 protocol configurations under infra.</td>
</tr>
<tr>
<td>access-protocol-l2</td>
<td>Used for Layer 2 protocol configurations under infra.</td>
</tr>
<tr>
<td>access-protocol-l3</td>
<td>Used for Layer 3 protocol configurations under infra.</td>
</tr>
<tr>
<td>access-protocol-mgmt</td>
<td>Used for fabric-wide policies for NTP, SNMP, DNS, and image management.</td>
</tr>
</tbody>
</table>
## AAA RBAC Roles and Privileges

### Role: access-admin

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access-protocol-ops</td>
<td>Used for operations-related access policies such as cluster policy and firmware policies.</td>
</tr>
<tr>
<td>access-qos</td>
<td>Used for changing CoPP and QoS-related policies.</td>
</tr>
</tbody>
</table>

### Role: fabric-admin

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fabric-connectivity-l1</td>
<td>Used for Layer 1 configuration under the fabric. Example: selectors and port Layer 1 policy and vPC protection.</td>
</tr>
<tr>
<td>fabric-connectivity-l2</td>
<td>Used in firmware and deployment policies for raising warnings for estimating policy deployment impact.</td>
</tr>
<tr>
<td>fabric-connectivity-mgmt</td>
<td>Used for atomic counter and diagnostic policies on leaf switches and spine switches.</td>
</tr>
<tr>
<td>fabric-connectivity-util</td>
<td>Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.</td>
</tr>
<tr>
<td>fabric-equipment</td>
<td>Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.</td>
</tr>
<tr>
<td>fabric-protocol-l1</td>
<td>Used for Layer 1 protocol configurations under the fabric.</td>
</tr>
<tr>
<td>fabric-protocol-l2</td>
<td>Used for Layer 2 protocol configurations under the fabric.</td>
</tr>
<tr>
<td>fabric-protocol-l3</td>
<td>Used for Layer 3 protocol configurations under the fabric.</td>
</tr>
<tr>
<td>fabric-protocol-ops</td>
<td>Used for ERSPAN and health score policies.</td>
</tr>
<tr>
<td>fabric-protocol-util</td>
<td>Used for firmware management traceroute and endpoint tracking policies.</td>
</tr>
<tr>
<td>tenant-connectivity-util</td>
<td>Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.</td>
</tr>
<tr>
<td>tenant-connectivity-l2</td>
<td>Used for Layer 2 connectivity changes, including bridge domains and subnets.</td>
</tr>
<tr>
<td>tenant-connectivity-l3</td>
<td>Used for Layer 3 connectivity changes, including VRFs.</td>
</tr>
<tr>
<td>tenant-protocol-ops</td>
<td>Used for tenant traceroute policies.</td>
</tr>
<tr>
<td>Role</td>
<td>Privilege</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
</tr>
<tr>
<td>nw-svc-admin</td>
<td>nw-svc-device</td>
</tr>
<tr>
<td></td>
<td>nw-svc-devshare</td>
</tr>
<tr>
<td></td>
<td>nw-svc-policy</td>
</tr>
<tr>
<td>nw-svc-params</td>
<td>nw-svc-params</td>
</tr>
</tbody>
</table>

**Role: ops**

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ops</td>
<td>Used for viewing the policies configured including troubleshooting policies.</td>
</tr>
</tbody>
</table>

*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 APIC.

**Role: read-all**

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access-connectivity-l1</td>
<td>Used for Layer 1 configuration under infra. Example: selectors and port Layer 1 policy configurations.</td>
</tr>
<tr>
<td>access-connectivity-l2</td>
<td>Used for Layer 2 configuration under infra. Example: Encap configurations on selectors, and attachable entity.</td>
</tr>
<tr>
<td>access-connectivity-l3</td>
<td>Used for Layer 3 configuration under infra and static route configurations under a tenant’s L3Out.</td>
</tr>
<tr>
<td>access-connectivity-mgmt</td>
<td>Used for management infra policies.</td>
</tr>
<tr>
<td>access-connectivity-util</td>
<td>Used for tenant ERSPAN policies.</td>
</tr>
<tr>
<td>access-equipment</td>
<td>Used for access port configuration.</td>
</tr>
<tr>
<td>access-protocol-l1</td>
<td>Used for Layer 1 protocol configurations under infra.</td>
</tr>
<tr>
<td>access-protocol-l2</td>
<td>Used for Layer 2 protocol configurations under infra.</td>
</tr>
<tr>
<td>access-protocol-l3</td>
<td>Used for Layer 3 protocol configurations under infra.</td>
</tr>
<tr>
<td>access-protocol-mgmt</td>
<td>Used for fabric-wide policies for NTP, SNMP, DNS, and image management.</td>
</tr>
</tbody>
</table>
### Role: read-all

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access-protocol-ops</td>
<td>Used for operations-related access policies such as cluster policy and firmware policies.</td>
</tr>
<tr>
<td>access-qos</td>
<td>Used for changing CoPP and QoS-related policies.</td>
</tr>
<tr>
<td>fabric-connectivity-l1</td>
<td>Used for Layer 1 configuration under the fabric. Example: selectors and port Layer 1 policy and vPC protection.</td>
</tr>
<tr>
<td>fabric-connectivity-l2</td>
<td>Used in firmware and deployment policies for raising warnings for estimating policy deployment impact.</td>
</tr>
<tr>
<td>fabric-protocol-l1</td>
<td>Used for Layer 1 protocol configurations under the fabric.</td>
</tr>
<tr>
<td>fabric-protocol-l2</td>
<td>Used for Layer 2 protocol configurations under the fabric.</td>
</tr>
<tr>
<td>fabric-protocol-l3</td>
<td>Used for Layer 3 protocol configurations under the fabric.</td>
</tr>
<tr>
<td>nw-svc-device</td>
<td>Used for managing Layer 4 to Layer 7 service devices.</td>
</tr>
<tr>
<td>nw-svc-devshare</td>
<td>Used for managing shared Layer 4 to Layer 7 service devices.</td>
</tr>
<tr>
<td>nw-svc-params</td>
<td>Used for managing Layer 4 to Layer 7 service policies.</td>
</tr>
<tr>
<td>nw-svc-policy</td>
<td>Used for managing Layer 4 to Layer 7 network service orchestration.</td>
</tr>
<tr>
<td>ops</td>
<td>Used for viewing the policies configured including troubleshooting policies.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> The <strong>ops</strong> role cannot be used for creating new monitoring and troubleshooting policies. Those policies need to be created by using the <strong>admin</strong> privilege, just like any other configurations in the APIC.</td>
</tr>
<tr>
<td>tenant-connectivity-util</td>
<td>Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.</td>
</tr>
<tr>
<td>tenant-connectivity-l2</td>
<td>Used for Layer 2 connectivity changes, including bridge domains and subnets.</td>
</tr>
<tr>
<td>tenant-connectivity-l3</td>
<td>Used for Layer 3 connectivity changes, including VRFs.</td>
</tr>
<tr>
<td>tenant-connectivity-mgmt</td>
<td>Used for tenant in-band and out-of-band management connectivity configurations and for debugging/monitoring policies such as atomic counters and health score.</td>
</tr>
<tr>
<td>tenant-epg</td>
<td>Used for managing tenant configurations such as deleting/creating endpoint groups, VRFs, and bridge domains.</td>
</tr>
<tr>
<td>Privilege</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tenant-ext-connectivity-l1</td>
<td>Used for write access firmware policies.</td>
</tr>
<tr>
<td>tenant-ext-connectivity-l2</td>
<td>Used for managing tenant L2Out configurations.</td>
</tr>
<tr>
<td>tenant-ext-connectivity-l3</td>
<td>Used for managing tenant L3Out configurations.</td>
</tr>
<tr>
<td>tenant-ext-connectivity-mgmt</td>
<td>Used as write access for firmware policies.</td>
</tr>
<tr>
<td>tenant-ext-connectivity-util</td>
<td>Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.</td>
</tr>
<tr>
<td>tenant-ext-protocol-l1</td>
<td>Used for managing tenant external Layer 1 protocols. Generally only used for write access for firmware policies.</td>
</tr>
<tr>
<td>tenant-ext-protocol-l2</td>
<td>Used for managing tenant external Layer 2 protocols. Generally only used for write access for firmware policies.</td>
</tr>
<tr>
<td>tenant-ext-protocol-l3</td>
<td>Used for managing tenant external Layer 3 protocols such as BGP, OSPF, PIM, and IGMP.</td>
</tr>
<tr>
<td>tenant-ext-protocol-mgmt</td>
<td>Used as write access for firmware policies.</td>
</tr>
<tr>
<td>tenant-ext-protocol-util</td>
<td>Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.</td>
</tr>
<tr>
<td>tenant-network-profile</td>
<td>Used for managing tenant configurations, such as deleting and creating network profiles, and deleting and creating endpoint groups.</td>
</tr>
<tr>
<td>tenant-protocol-l1</td>
<td>Used for managing configurations for Layer 1 protocols under a tenant.</td>
</tr>
<tr>
<td>tenant-protocol-l2</td>
<td>Used for managing configurations for Layer 2 protocols under a tenant.</td>
</tr>
<tr>
<td>tenant-protocol-l3</td>
<td>Used for managing configurations for Layer 3 protocols under a tenant.</td>
</tr>
<tr>
<td>tenant-protocol-mgmt</td>
<td>Only used as write access for firmware policies.</td>
</tr>
<tr>
<td>tenant-protocol-ops</td>
<td>Used for tenant traceroute policies.</td>
</tr>
<tr>
<td>tenant-QoS</td>
<td>Used for QoS-related configurations for a tenant.</td>
</tr>
<tr>
<td>tenant-security</td>
<td>Used for contract-related configurations for a tenant.</td>
</tr>
<tr>
<td>vmm-connectivity</td>
<td>Used to read all the objects in APIC’s VMM inventory required for virtual machine connectivity.</td>
</tr>
<tr>
<td>vmm-ep</td>
<td>Used to read virtual machine and hypervisor endpoints in the APIC’s VMM inventory.</td>
</tr>
<tr>
<td>vmm-policy</td>
<td>Used for managing policies for virtual machine networking.</td>
</tr>
<tr>
<td>vmm-protocol-ops</td>
<td>Not used by VMM policies.</td>
</tr>
</tbody>
</table>
### AAA RBAC Roles and Privileges

<table>
<thead>
<tr>
<th>Role: read-all</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmm-security</td>
<td>Used for managing authentication policies for VMM, such as the username and password for VMware vCenter.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Role: tenant-admin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa</td>
<td>Used for configuring authentication, authorization, accounting and import/export policies.</td>
</tr>
<tr>
<td>access-connectivity-l1</td>
<td>Used for Layer 1 configuration under infra. Example: selectors and port Layer 1 policy configurations.</td>
</tr>
<tr>
<td>access-connectivity-l2</td>
<td>Used for Layer 2 configuration under infra. Example: Encap configurations on selectors, and attachable entity.</td>
</tr>
<tr>
<td>access-connectivity-l3</td>
<td>Used for Layer 3 configuration under infra and static route configurations under a tenant's L3Out.</td>
</tr>
<tr>
<td>access-connectivity-mgmt</td>
<td>Used for management infra policies.</td>
</tr>
<tr>
<td>access-connectivity-util</td>
<td>Used for tenant ERSPAN policies.</td>
</tr>
<tr>
<td>access-equipment</td>
<td>Used for access port configuration.</td>
</tr>
<tr>
<td>access-protocol-l1</td>
<td>Used for Layer 1 protocol configurations under infra.</td>
</tr>
<tr>
<td>access-protocol-l2</td>
<td>Used for Layer 2 protocol configurations under infra.</td>
</tr>
<tr>
<td>access-protocol-l3</td>
<td>Used for Layer 3 protocol configurations under infra.</td>
</tr>
<tr>
<td>access-protocol-mgmt</td>
<td>Used for fabric-wide policies for NTP, SNMP, DNS, and image management.</td>
</tr>
<tr>
<td>access-protocol-ops</td>
<td>Used for operations-related access policies such as cluster policy and firmware policies.</td>
</tr>
<tr>
<td>access-qos</td>
<td>Used for changing CoPP and QoS-related policies.</td>
</tr>
<tr>
<td>fabric-connectivity-l1</td>
<td>Used for Layer 1 configuration under the fabric. Example: selectors and port Layer 1 policy and vPC protection.</td>
</tr>
<tr>
<td>fabric-connectivity-l2</td>
<td>Used in firmware and deployment policies for raising warnings for estimating policy deployment impact.</td>
</tr>
<tr>
<td>fabric-connectivity-mgmt</td>
<td>Used for atomic counter and diagnostic policies on leaf switches and spine switches.</td>
</tr>
</tbody>
</table>
### Role: tenant-admin

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fabric-connectivity-util</td>
<td>Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.</td>
</tr>
<tr>
<td>fabric-equipment</td>
<td>Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.</td>
</tr>
<tr>
<td>fabric-protocol-l1</td>
<td>Used for Layer 1 protocol configurations under the fabric.</td>
</tr>
<tr>
<td>fabric-protocol-l2</td>
<td>Used for Layer 2 protocol configurations under the fabric.</td>
</tr>
<tr>
<td>fabric-protocol-l3</td>
<td>Used for Layer 3 protocol configurations under the fabric.</td>
</tr>
<tr>
<td>fabric-protocol-ops</td>
<td>Used for ERSPAN and health score policies.</td>
</tr>
<tr>
<td>fabric-protocol-util</td>
<td>Used for firmware management traceroute and endpoint tracking policies.</td>
</tr>
<tr>
<td>nw-svc-device</td>
<td>Used for managing Layer 4 to Layer 7 service devices.</td>
</tr>
<tr>
<td>nw-svc-devshare</td>
<td>Used for managing shared Layer 4 to Layer 7 service devices.</td>
</tr>
<tr>
<td>nw-svc-params</td>
<td>Used for managing Layer 4 to Layer 7 service policies.</td>
</tr>
<tr>
<td>nw-svc-policy</td>
<td>Used for managing Layer 4 to Layer 7 network service orchestration.</td>
</tr>
<tr>
<td>ops</td>
<td>Used for viewing the policies configured including troubleshooting policies.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> The <strong>ops</strong> role cannot be used for creating new monitoring and troubleshooting policies. Those policies need to be created by using the <strong>admin</strong> privilege, just like any other configurations in the APIC.</td>
</tr>
<tr>
<td>tenant-connectivity-util</td>
<td>Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.</td>
</tr>
<tr>
<td>tenant-connectivity-l2</td>
<td>Used for Layer 2 connectivity changes, including bridge domains and subnets.</td>
</tr>
<tr>
<td>tenant-connectivity-l3</td>
<td>Used for Layer 3 connectivity changes, including VRFs.</td>
</tr>
<tr>
<td>tenant-connectivity-mgmt</td>
<td>Used for tenant in-band and out-of-band management connectivity configurations and for debugging/monitoring policies such as atomic counters and health score.</td>
</tr>
<tr>
<td>tenant-epg</td>
<td>Used for managing tenant configurations such as deleting/creating endpoint groups, VRFs, and bridge domains.</td>
</tr>
<tr>
<td>Role: tenant-admin</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>tenant-ext-connectivity-l1</td>
<td>Used for write access firmware policies.</td>
</tr>
<tr>
<td>tenant-ext-connectivity-l2</td>
<td>Used for managing tenant L2Out configurations.</td>
</tr>
<tr>
<td>tenant-ext-connectivity-l3</td>
<td>Used for managing tenant L3Out configurations.</td>
</tr>
<tr>
<td>tenant-ext-connectivity-mgmt</td>
<td>Used as write access for firmware policies.</td>
</tr>
<tr>
<td>tenant-ext-connectivity-util</td>
<td>Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.</td>
</tr>
<tr>
<td>tenant-ext-protocol-l1</td>
<td>Used for managing tenant external Layer 1 protocols. Generally only used for write access for firmware policies.</td>
</tr>
<tr>
<td>tenant-ext-protocol-l2</td>
<td>Used for managing tenant external Layer 2 protocols. Generally only used for write access for firmware policies.</td>
</tr>
<tr>
<td>tenant-ext-protocol-l3</td>
<td>Used for managing tenant external Layer 3 protocols such as BGP, OSPF, PIM, and IGMP.</td>
</tr>
<tr>
<td>tenant-ext-protocol-mgmt</td>
<td>Used as Write access for firmware policies.</td>
</tr>
<tr>
<td>tenant-ext-protocol-util</td>
<td>Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.</td>
</tr>
<tr>
<td>tenant-network-profile</td>
<td>Used for managing tenant configurations, such as deleting and creating network profiles, and deleting and creating endpoint groups.</td>
</tr>
<tr>
<td>tenant-protocol-l1</td>
<td>Used for managing configurations for Layer 1 protocols under a tenant.</td>
</tr>
<tr>
<td>tenant-protocol-l2</td>
<td>Used for managing configurations for Layer 2 protocols under a tenant.</td>
</tr>
<tr>
<td>tenant-protocol-l3</td>
<td>Used for managing configurations for Layer 3 protocols under a tenant.</td>
</tr>
<tr>
<td>tenant-protocol-mgmt</td>
<td>Only used as write access for firmware policies.</td>
</tr>
<tr>
<td>tenant-protocol-ops</td>
<td>Used for tenant traceroute policies.</td>
</tr>
<tr>
<td>tenant-QoS</td>
<td>Used for QoS-related configurations for a tenant.</td>
</tr>
<tr>
<td>tenant-security</td>
<td>Used for contract-related configurations for a tenant.</td>
</tr>
<tr>
<td>vmm-connectivity</td>
<td>Used to read all the objects in APIC’s VMM inventory required for virtual machine connectivity.</td>
</tr>
<tr>
<td>vmm-ep</td>
<td>Used to read virtual machine and hypervisor endpoints in the APIC’s VMM inventory.</td>
</tr>
<tr>
<td>vmm-policy</td>
<td>Used for managing policies for virtual machine networking.</td>
</tr>
<tr>
<td>vmm-protocol-ops</td>
<td>Not used by VMM policies.</td>
</tr>
</tbody>
</table>
## Access, Authentication, and Accounting

### AAA RBAC Roles and Privileges

#### Role: tenant-admin

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmm-security</td>
<td>Used for managing authentication policies for VMM, such as the username and password for VMware vCenter.</td>
</tr>
</tbody>
</table>

#### Role: tenant-ext-admin

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

<table>
<thead>
<tr>
<th>Service</th>
<th>Connectivity</th>
<th>QoS</th>
<th>Security</th>
<th>Application</th>
<th>Fault</th>
<th>Stats</th>
<th>Provider</th>
<th>Service Profile</th>
<th>Service Chain</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMM</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fabric</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>External</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tenant</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>EPG, NP</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Infra</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ops</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Service</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Selectively Expose Physical Resources across Security Domains**

A fabric-wide administrator uses RBAC rules to selectively expose physical resources to users that otherwise are inaccessible because they are in a different security domain.

For example, if a user in tenant Solar needs access to a virtual machine management (VMM) domain, the fabric-wide admin could create an RBAC rule to allow this. The RBAC rule is comprised of these two parts: the distinguished name (DN) that locates the object to be accessed plus the name of the security domain that contains the user who will access the object. So, in this example, when designated users in the security domain Solar are logged in, this rule gives them access to the VMM domain as well as all its child objects in the tree. To give users in multiple security domains access to the VMM domain, the fabric-wide administrator would create an RBAC rule for each security domain that contains the DN for the VMM domain plus the security domain.

**Note**

While an RBAC rule exposes an object to a user in a different part of the management information tree, it is not possible to use the CLI to navigate to such an object by traversing the structure of the tree. However, as long as the user knows the DN of the object included in the RBAC rule, the user can use the CLI to locate it via an MO find command.

**Enable Sharing of Services across Security Domains**

A fabric-wide administrator uses RBAC rules to provision trans-tenant EPG communications that enable shared services across tenants.
APIC Local Users

An administrator can choose not to use external AAA servers but rather configure users on the APIC itself. These users are called APIC-local users.

At the time a user sets their password, the APIC validates it against the following criteria:

- Minimum password length is 8 characters.
- Maximum password length is 64 characters.
- Has fewer than three consecutive repeated characters.
- Must have characters from at least three of the following characters types: lowercase, uppercase, digit, symbol.
- Does not use easily guessed passwords.
- Cannot be the username or the reverse of the username.
- Cannot be any variation of cisco, isco or any permutation of these characters or variants obtained by changing the capitalization of letters therein.

The APIC also enables administrators to grant access to users configured on externally managed authentication Lightweight Directory Access Protocol (LDAP), RADIUS, TACACS+, or SAML servers. Users can belong to different authentication systems and can log in simultaneously to the APIC.

In addition, OTP can be enabled for a Local User which is a one-time password that changes every 30 seconds. Once OTP is enabled, APIC generates a random human readable 16 binary octet that are base32 OTP Key. This OTP Key is used to generate OTP for the user.

The following figure shows how the process works for configuring an admin user in the local APIC authentication database who has full access to the entire ACI fabric.

*Figure 1: APIC Local User Configuration Process*
The security domain “all” represents the entire Managed Information Tree (MIT). This domain includes all policies in the system and all nodes managed by the APIC. Tenant domains contain all the users and managed objects of a tenant. Tenant administrators should not be granted access to the “all” domain.

The following figure shows the access that the admin user Joe Stratus has to the system.

**Figure 2: Result of Configuring Admin User for “all” Domain**

The user Joe Stratus with read-write “admin” privileges is assigned to the domain “all” which gives him full access to the entire system.

**Externally Managed Authentication Server Users**

The following figure shows how the process works for configuring an admin user in an external RADIUS server who has full access to the tenant Solar.
The following figure shows the access the admin user Jane Cirrus has to the system.
In this example, the Solar tenant administrator has full access to all the objects contained in the Solar tenant as well as read-only access to the tenant Common. Tenant admin Jane Cirrus has full access to the tenant Solar, including the ability to create new users in tenant Solar. Tenant users are able to modify configuration parameters of the ACI fabric that they own and control. They also are able to read statistics and monitor faults and events for the entities (managed objects) that apply to them such as endpoints, endpoint groups (EPGs) and application profiles.

In the example above, the user Jane Cirrus was configured on an external RADIUS authentication server. To configure an AV Pair on an external authentication server, add a Cisco AV Pair to the existing user record. The Cisco AV Pair specifies the Role-Based Access Control (RBAC) roles and privileges for the user on the APIC. The RADIUS server then propagates the user privileges to the APIC controller.

In the example above, the configuration for an open radius server (/etc/raddb/users) is as follows:

```
janecirrus Cleartext-Password := "<password>"
Cisco-avpair = "shell:domains = solar/admin,,common//read-all(16001)"
```

This example includes the following elements:

- **janecirrus** is the tenant administrator
- **solar** is the tenant
- **admin** is the role with write privileges
**Cisco AV Pair Format**

The Cisco APIC requires that an administrator configure a Cisco AV Pair on an external authentication server and only looks for one AV pair string. To do so, an administrator adds a Cisco AV pair to the existing user record. The Cisco AV pair specifies the APIC required RBAC roles and privileges for the user.

In order for the AV pair string to work, it must be formatted as follows:

```
shell:domains =
ACI_Security_Domain_1/ACI_Write_Role_1|ACI_Write_Role_2|ACI_Write_Role_3/ACI_Read_Role_1|ACI_Read_Role_2,
ACI_Security_Domain_2/ACI_Write_Role_1|ACI_Write_Role_2|ACI_Write_Role_3/ACI_Read_Role_1|ACI_Read_Role_2,
ACI_Security_Domain_3/ACI_Write_Role_1|ACI_Write_Role_2|ACI_Write_Role_3/ACI_Read_Role_1|ACI_Read_Role_2
```

- **shell:domains** - Required so that ACI reads the string correctly. This must always prepend the shell string.
- **ACI_Security_Domain_1/admin** - Grants admin read only access to the tenants in this security domain.
- **ACI_Security_Domain_2/admin** - Grants admin write access to the tenants in this security domain.
- **ACI_Security_Domain_3/read-all** - Grants read-all write access to the tenants in this security domain.

Note

/`s separate the security domain, write, read sections of the string. `/`s separate multiple write or read roles within the same security domain.

Note

Starting with Cisco APIC release 2.1, if no UNIX ID is provided in AV Pair, the APIC allocates the unique UNIX user ID internally.

The APIC supports the following regexes:

```
shell:domains\\s*\[=:\]\s*((\\S+?/\\S*?/\\S*?)\s*\{0,31\})\s*\(\d+\)\\$  
shell:domains\\s*\[=:\]\s*((\\S+?/\\S*?/\\S*?)\s*\{0,31\})\s*\(\d+\)\\$  
```

**Examples:**

- Example 1: A Cisco AV Pair that contains a single Login domain with only writeRoles:

  ```
  shell:domains=ACI_Security_Domain_1/Write_Role_1/Write_Role_2/
  ```

- Example 2: A Cisco AV Pair that contains a single Login domain with only readRoles:

  ```
  shell:domains=Security_Domain_1//Read_Role_1|Read_Role_2
  ```
The "/" character is a separator between writeRoles and readRoles per Login domain and is required even if
only one type of role is to be used.

The Cisco AVpair string is case sensitive. Although a fault may not be seen, using mismatching cases for the
domain name or roles could lead to unexpected privileges being given.

**AV Pair GUI Configuration**

The security domain is defined in the ACI GUI under **Admin > AAA > Security Management > Security
Domains** and assigned to a tenant under **Tenants > Tenant_Name > Policy**.

A security domain must have either a read or write role. These roles are defined in **APIC > Admin > Security
Management > Roles**. If a role is input into the write section it automatically grants read privileges of the
same level so there is no need to have ACI_Security_Domain_1/admin/admin.

**Change Remote User Role**

User-privileges can be modified “dynamically”, which allows the user to request for a role-change, and is
allowed or denied the requested role based on information stored locally or remotely.

The role-change is only supported through the Cisco ACS server and can be done by role assignment based
on explicit "request".

The ACI fabric supports external authentication using Radius, TACACS+ and LDAP protocols. Both the
above-mentioned methods assume that the remote authentication server has components to support the
role-change functionality.

The Cisco Secure ACS server provides the remote authentication, authorization and accounting features for
the TACACS+ protocol.

Rules are matched, either with **Default Device Admin** or **Default Network Access Service**.

In the Authorization, another set of rules are configured:

- **AVPairOps**: matches the tacacs+ username and AVPair value (cisco-av-pair*newrole). If the rule
matches, the ACI_OPS shell-profile is returned

- **NoAVPair**: matches only the tacacs+ username and return ACI_ADMIN shell profile on match

- **opsuser**: matches only the protocol and returns ACI_OPS shell profile

**Change the Remote User Role Using the GUI**

**Before you begin**

Roles must first be configured on the Cisco ASC Server to match the AVPairs and selected
shell-authorization-profile based on the match.

**Procedure**

**Step 1** Create an ASC Authorization Policy navigate to **Access Policies > Access Services > Default Device Admin
Identity** and perform the following steps:
Shell Profile is configured with CiscoAVPair, which is used to Authorize the User.

a) Add the condition to TACACS+:AVPair equals cisco-av-pair* and click OK.
   
   Note: The user is authorized with the cisco-av-pair role by default.

b) Add the condition to TACACS+:AVPair equals cisco-av-pair*readall and click OK.
   
   Note: The keyword readall is used in APIC to change the Role from default Role to readall Role (read-all is configured in Shell-Profile).

Step 2 Log in to the APIC GUI, click the welcome, <login_name> drop-down list and choose Change Remote User Role.

Step 3 In the Change Remote User Role dialog box, enter the information in the User Name, Password, and New Role fields and click Submit.
   
   The GUI will refresh with the new role applied.

   Note: To return to the parent role, open the Change Remote User Role dialog box again and enter the information for User Name and Password but leave the New Role field blank.

---

Change the Remote User Role Using REST API

Before you begin

Roles must first be configured on the Cisco ASC Server to match the AVPairs and selected shell-authorization-profile based on the match.

The user logs in with the user-name apicadmin and password.

Procedure

Step 1 Change to a new role:

Example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<aaaChangeRole>
  <attributes userName="apic#tacacs" apicadmin="pwd Ins3965!" role="newrole"/>
</aaaChangeRole>
```

Step 2 Return to the original role:

Example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<aaaChangeRole>
  <attributes userName="apic#tacacs" apicadmin="pwd Ins3965!" role=""/>
</aaaChangeRole>
```

---

About Signature-Based Transactions

The APIC controllers in a Cisco ACI fabric offer different methods to authenticate users.
The primary authentication method uses a username and password and the APIC REST API returns an authentication token that can be used for future access to the APIC. This may be considered insecure in a situation where HTTPS is not available or enabled.

Another form of authentication that is offered utilizes a signature that is calculated for every transaction. The calculation of that signature uses a private key that must be kept secret in a secure location. When the APIC receives a request with a signature rather than a token, the APIC utilizes an X.509 certificate to verify the signature. In signature-based authentication, every transaction to the APIC must have a newly calculated signature. This is not a task that a user should do manually for each transaction. Ideally this function should be utilized by a script or an application that communicates with the APIC. This method is the most secure as it requires an attacker to crack the RSA/DSA key to forge or impersonate the user credentials.

Additionally, you must use HTTPS to prevent replay attacks.

Before you can use X.509 certificate-based signatures for authentication, verify that the following pre-requisite tasks are completed:

1. Create an X.509 certificate and private key using OpenSSL or a similar tool.
2. Create a local user on the APIC. (If a local user is already available, this task is optional).
3. Add the X.509 certificate to the local user on the APIC.

Guidelines and Limitations

Follow these guidelines and limitations:

- Local users are supported. Remote AAA users are not supported.
- The APIC GUI does not support the certificate authentication method.
- WebSockets and eventchannels do not work for X.509 requests.
- Certificates signed by a third party are not supported. Use a self-signed certificate.

Accounting

ACI fabric accounting is handled by these two managed objects (MO) that are processed by the same mechanism as faults and events:

- The `aaaSessionLR` MO tracks user account login and logout sessions on the APIC and switches, and token refresh. The ACI fabric session alert feature stores information such as the following:
  - Username
  - IP address initiating the session
  - Type (telnet, https, REST etc.)
  - Session time and length
  - Token refresh – a user account login event generates a valid active token which is required in order for the user account to exercise its rights in the ACI fabric.
Token expiration is independent of login; a user could log out but the token expires according to the duration of the timer value it contains.

- The aaaModLR MO tracks the changes users make to objects and when the changes occurred.
- If the AAA server is not pingable, it is marked unavailable and a fault is seen.

Both the aaaSessionLR and aaaModLR event logs are stored in APIC shards. Once the data exceeds the pre-set storage allocation size, it overwrites records on a first-in first-out basis.

In the event of a destructive event such as a disk crash or a fire that destroys an APIC cluster node, the event logs are lost; event logs are not replicated across the cluster.

The aaaModLR and aaaSessionLR MOs can be queried by class or by distinguished name (DN). A class query provides all the log records for the whole fabric. All aaaModLR records for the whole fabric are available from the GUI at the Fabric > Inventory > POD > History > Audit Log section. The APIC GUI History > Audit Log options enable viewing event logs for a specific object identified in the GUI.

The standard syslog, callhome, REST query, and CLI export mechanisms are fully supported for aaaModLR and aaaSessionLR MO query data. There is no default policy to export this data.

There are no pre-configured queries in the APIC that report on aggregations of data across a set of objects or for the entire system. A fabric administrator can configure export policies that periodically export aaaModLR and aaaSessionLR query data to a syslog server. Exported data can be archived periodically and used to generate custom reports from portions of the system or across the entire set of system logs.

## Routed Connectivity to External Networks as a Shared Service Billing and Statistics

The APIC can be configured to collect byte count and packet count billing statistics from a port configured for routed connectivity to external networks (an l3extInstP EPG) as a shared service. Any EPG in any tenant can share an l3extInstP EPG for routed connectivity to external networks. Billing statistics can be collected for each EPG in any tenant that uses an l3extInstP EPG as a shared service. The leaf switch where the l3extInstP is provisioned forwards the billing statistics to the APIC where they are aggregated. Accounting policies can be configured to periodically export these billing statistics to a server.

## Configuration

### Configuring a Local User

In the initial configuration script, the admin account is configured and the admin is the only user when the system starts. The APIC supports a granular, role-based access control system where user accounts can be created with various roles including non-admin users with fewer privileges.
Configuring a Local User Using the GUI

Before you begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- As appropriate, the security domain(s) that the user will access are defined. For example, if the new use account will be restricted to accessing a tenant, the tenant domain is tagged accordingly.
- An APIC user account is available that will enable the following:
  - Creating the TACACS+ and TACACS+ provider group.
  - Creating the local user account in the target security domain(s). If the target domain is `all`, the login account used to create the new local user must be a fabric-wide administrator that has access to `all`. If the target domain is a tenant, the login account used to create the new local user must be a tenant administrator that has full read write access rights to the target tenant domain.

Procedure

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>On the menu bar, choose <strong>ADMIN &gt; AAA</strong>.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>In the <strong>Navigation</strong> pane, click <strong>AAA Authentication</strong>.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>In the <strong>Work</strong> pane, verify that in the default <strong>Authentication</strong> field, the <strong>Realm</strong> field displays as Local.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>In the <strong>Navigation</strong> pane, expand <strong>Security Management &gt; Local Users</strong>. The admin user is present by default.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>In the <strong>Navigation</strong> pane, right-click <strong>Create Local User</strong>.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>In the <strong>User Identity</strong> dialog box, enter a <strong>Login ID</strong> and <strong>Password</strong> for the user, and click <strong>Next</strong>.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>In the <strong>Security</strong> dialog box, choose the desired security domain for the user, and click <strong>Next</strong>.</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>In the <strong>Roles</strong> dialog box, click the radio buttons to choose the roles for your user, and click <strong>Next</strong>. You can provide read-only or read/write privileges.</td>
</tr>
</tbody>
</table>
| **Step 9** | In the **User Identity** dialog box, perform the following actions:  
  a) In the **Login ID** field, add an ID.  
  b) In the **Password** field, enter the password.  
  At the time a user sets their password, the APIC validates it against the following criteria:  
    - Minimum password length is 8 characters.  
    - Maximum password length is 64 characters.  
    - Has fewer than three consecutive repeated characters.  
    - Must have characters from at least three of the following characters types: lowercase, uppercase, digit, symbol.  
    - Does not use easily guessed passwords.  
    - Cannot be the username or the reverse of the username. |
Configuring SSH Public Key Authentication Using the GUI

Before you begin

• Create a local user account in the target security domain(s). If the target domain is `all`, the login account used to create the new local user must be a fabric-wide administrator that has access to `all`. If the target domain is a tenant, the login account used to create the new local user must be a tenant administrator that has full read write access rights to the target tenant domain.

• Generate a public key using the Unix command `ssh-keygen`.
  The default login domain must be set to `local`.

Procedure

Step 1  On the menu bar, choose **ADMIN > Security Management > Local Users**.

Step 2  In the **Navigation** pane, click the name of the user that you previously created.

Step 3  In the **Work** pane, expand the **SSH Keys** table, and insert the following information:
  a) In the **Name** field, enter a name for the key.
  b) In the **Key** field, insert the public key previously created. Click **Update**.

  **Note**  To create the SSH Private Key File for downloading to a remote location then in the menu bar, expand **Firmware > Download Tasks**.

Configuring a Local User Using the NX-OS Style CLI

Procedure

Step 1  In the NX-OS CLI, start in configuration mode, shown as follows:

  **Example:**

  ```
  apic1# configure
  apic1(config)#
  ```
Step 2  Create a new user, shown as follows:

Example:

```
api1(config)# username
WORD          User name (Max Size 28)
admin
cli-user
jigarshah
test1
testUser

api1(config)# username test
api1(config-username)#
account-status  Set The status of the locally-authenticated user account.
certificate     Create AAA user certificate in X.509 format.
clear-pwd-history  Clears the password history of a locally-authenticated user
domain         Create the AAA domain to which the user belongs.
email           Set The email address of the locally-authenticated user.
exit            Exit from current mode
expiration      If expires enabled, Set expiration date of locally-authenticated user
account.
expires         Enable expiry for locally-authenticated user account
fabric         show fabric related information
first-name      Set the first name of the locally-authenticated user.
last-name       Set The last name of the locally-authenticated user.
no              Negate a command or set its defaults
password        Set The system user password.
phone           Set The phone number of the locally-authenticated user.
pwd-lifetime    Set The lifetime of the locally-authenticated user password.
pwd-strength-check  Enforces the strength of the user password
show           Show running system information
ssh-key        Update ssh key for the user for ssh authentication
where          show the current mode

api1(config-username)# exit
```

---

**Configuring a Local User Using the REST API**

**Procedure**

Create a local user.

**Example:**

**URL:** https://apic-ip-address/api/policymgr/mo/uni/userext.xml

**POST CONTENT:**

```xml
<aaaUser name="operations" phone="" pwd="<strong_password>" >
  <aaaUserDomain childAction="" descr="" name="all" rn="userdomain-all" status="">
    <aaaUserRole childAction="" descr="" name="Ops" privType="writePriv"/>
  </aaaUserDomain>
</aaaUser>
```

---

---
Generating an X.509 Certificate and a Private Key

Procedure

Step 1
Enter an OpenSSL command to generate an X.509 certificate and private key.

Example:
```
$ openssl req -new -newkey rsa:1024 -days 36500 -nodes -x509 -keyout userabc.key -out userabc.crt -subj '/CN=User ABC/O=Cisco Systems/C=US'
```

Note
- Once the X.509 certificate is generated, it will be added to the users profile on the APIC, and it is used to verify signatures. The private key is used by the client to generate the signatures.
- The certificate contains a public key but not the private key. The public key is the primary information used by the APIC to verify the calculated signature. The private key is never stored on the APIC. You must keep it secret.

Step 2
Display the fields in the certificate using OpenSSL.

Example:
```
$ openssl x509 -text -in userabc.crt
```

Certificate:
```
Data:
Version: 3 (0x2)
Signature Algorithm: sha1WithRSAEncryption
Issuer: CN=User ABC, O=Cisco Systems, C=US
Validity
    Not Before: Jan 12 16:36:14 2015 GMT
    Not After : Dec 19 16:36:14 2114 GMT
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:92:35:12:cd:2b:78:ef:9d:ca:0e:11:77:77:3a:
          ce:c4:7f:61:1d:3c:9e:ae:d8:88:be:80:a0:0a:90:
          3a:3f:68:0b:9c:7cc:cb:70:b9:c7:3f:e8:db:85:d8:
          5f:bc:35:d2:bl:07:bec:ec:e1
        Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Subject Key Identifier:
    X509v3 Authority Key Identifier:
      DirName:/CN=User ABC/O=Cisco Systems/C=US
X509v3 Basic Constraints:
    CA:TRUE
Signature Algorithm: sha1WithRSAEncryption
```

Creating a Local User and Adding a User Certificate Using the GUI

Procedure

**Step 1**
On the menu bar, choose **ADMIN > AAA**.

**Step 2**
In the **Navigation** pane, click **AAA Authentication**.

**Step 3**
In the **Work** pane, verify that in the default **Authentication** field, the **Realm** field displays as **Local**.

**Step 4**
In the **Navigation** pane, expand **Security Management > Local Users**. The admin user is present by default.

**Step 5**
In the **Navigation** pane, right-click **Local Users** and click **Create Local User**.

**Step 6**
In the **Security** dialog box, choose the desired security domain for the user, and click **Next**.

**Step 7**
In the **Roles** dialog box, click the radio buttons to choose the roles for your user, and click **Next**.

You can provide read-only or read/write privileges.

**Step 8**
In the **User Identity** dialog box, perform the following actions:

a) In the **Login ID** field, add an ID.

b) In the **Password** field, enter the password.

c) In the **Confirm Password** field, confirm the password.

d) Click **Finish**.

**Step 9**
In the **Navigation** pane, click the name of the user that you created. In the **Work** pane, expand the + sign next to your user in the **Security Domains** area. The access privileges for your user are displayed.

**Step 10**
In the **Work** pane, in the **User Certificates** area, click the user certificates + sign, and in the **Create X509 Certificate** dialog box, perform the following actions:

a) In the **Name** field, enter a certificate name.

b) In the **Data** field, enter the user certificate details.

c) Click **Submit**.

The X509 certificate is created for the local user.
Creating a Local User and Adding a User Certificate Using the REST API

Procedure

Create a local user and add a user certificate.

Example:

Method: POST

URL: http://apic/api/node/mo/uni/userext/user-userabc.json

Payload:

```json
{
    "aaaUser": {
        "attributes": {
            "name": "userabc",
            "firstName": "Adam",
            "lastName": "BC",
            "phone": "408-525-4766",
            "email": "userabc@cisco.com",
        },
        "children": [{
            "aaaUserCert": {
                "attributes": {
                    "name": "userabc.crt",
                    "data": "-----BEGIN CERTIFICATE-----
MIICjjCCAfegAwIBAgIJAMQnbE
<snipped content> ==
-----END CERTIFICATE-----",
                },
            },
            "aaaUserDomain": {
                "attributes": {
                    "name": "all",
                },
            },
            "aaaUserRole": {
                "attributes": {
                    "name": "aaa",
                    "privType": "writePriv",
                },
                "children": []
            }
        },
        "aaaUserRole": {
            "attributes": {
                "name": "access-admin",
                "privType": "writePriv",
            },
            "children": []
        }
    }
}
```

Access, Authentication, and Accounting
Creating a Local User and Adding a User Certificate Using the REST API

Access, Authentication, and Accounting

```json
{
  "children": [],
  "aaaUserRole": {
    "attributes": {
      "name": "nw-svc-admin",
      "privType": "writePriv",
    },
    "children": []
  },
  "aaaUserRole": {
    "attributes": {
      "name": "ops",
      "privType": "writePriv",
    },
    "children": []
  },
  "aaaUserRole": {
    "attributes": {
      "name": "read-all",
      "privType": "writePriv",
    },
    "children": []
  },
  "aaaUserRole": {
    "attributes": {
      "name": "tenant-admin",
      "privType": "writePriv",
    },
    "children": []
  },
  "aaaUserRole": {
    "attributes": {
      "name": "tenant-ext-admin",
      "privType": "writePriv",
    },
    "children": []
  },
  "aaaUserRole": {
    "attributes": {
      "name": "vmm-admin",
      "privType": "writePriv",
    },
    "children": []
  }
}
```
Creating a Local User Using Python SDK

Procedure

Create a local user.

Example:

```python
#!/usr/bin/env python
from cobra.model.pol import Uni as PolUni
from cobra.model.aaa import UserEp as AaaUserEp
from cobra.model.aaa import User as AaaUser
from cobra.model.aaa import UserCert as AaaUserCert
from cobra.model.aaa import UserDomain as AaaUserDomain
from cobra.model.aaa import UserRole as AaaUserRole
from cobra.mit.access import MoDirectory
from cobra.mit.session import LoginSession
from cobra.internal.codec.jsoncodec import toJSONStr

APIC = 'http://10.10.10.1'
username = 'admin'
password = 'p@$$w0rd'

session = LoginSession(APIC, username, password)
modir = MoDirectory(session)
modir.login()

def readFile(fileName=None, mode="r"):
    if fileName is None:
        return ""
    fileData = ""
    with open(fileName, mode) as aFile:
        fileData = aFile.read()
    return fileData

# Use a dictionary to define the domain and a list of tuples to define
# our aaaUserRoles (roleName, privType)
# This can further be abstracted by doing a query to get the valid
# roles, that is what the GUI does
userRoles = {
    'all': [("aaa", 'writePriv'),
            ("access-admin", 'writePriv'),
            ("admin", 'writePriv'),
            ("fabric-admin", 'writePriv'),
            ("nw-svc-admin", 'writePriv'),
            ("ops", 'writePriv'),
            ("read-all", 'writePriv'),
            ("tenant-admin", 'writePriv'),
            ("tenant-ext-admin", 'writePriv'),
            ("vmm-admin", 'writePriv')],
}

uni = PolUni('') # '' is the Dn string for topRoot
aaaUserEp = AaaUserEp(uni)
aaaUser = AaaUser(aaaUserEp, 'userabc', firstName='Adam',
                 email='userabc@cisco.com')

aaaUser.lastName = 'BC'
aaaUser.phone = '555-111-2222'
```
aaaUserCert = AaaUserCert(aaaUser, 'userabc.crt')
aaaUserCert.data = readFile("/tmp/userabc.crt")

# Now add each aaaUserRole to the aaaUserDomains which are added to the
# aaaUserCert
for domain,roles in userRoles.items():
    aaaUserDomain = AaaUserDomain(aaaUser, domain)
    for roleName, privType in roles:
        aaaUserRole = AaaUserRole(aaaUserDomain, roleName,
                                   privType=privType)
print toJSONStr(aaaUser, prettyPrint=True)

# End of Script to create a user

---

**Using a Private Key to Calculate a Signature**

**Before you begin**

You must have the following information available:

- HTTP method - GET, POST, DELETE
- REST API URI being requested, including any query options
- For POST requests, the actual payload being sent to the APIC
- The private key used to generate the X.509 certificate for the user
- The distinguished name for the user X.509 certificate on the APIC

**Procedure**

**Step 1**

Concatenate the HTTP method, REST API URI, and payload together in this order and save them to a file.

This concatenated data must be saved to a file for OpenSSL to calculate the signature. In this example, we use a filename of payload.txt. Remember that the private key is in a file called userabc.key.

**Example:**

GET example:

GET http://10.10.10.1/api/class/fvTenant.json?rsp-subtree=children

POST example:

POST http://10.10.10.1/api/mo/tn-test.json{"fvTenant": {"attributes": {"status": "deleted", "name": "test"}}}"

**Step 2**

Calculate a signature using the private key and the payload file using OpenSSL.

**Example:**

openssl dgst -sha256 -sign userabc.key payload.txt > payload_sig.bin

The resulting file has the signature printed on multiple lines.
Step 3  Strip the signature of the new lines using Bash.

**Example:**

```
$ tr -d \n < payload_sig.base64
P+OTqK0CeAZjl7+Gute2RlWw8OGgtzE0wsLlx8fIXX14V79Zl7
Ou8IdJH9CB4W6CEvdICxqkv3KaQazCIC0+Bn07o3qF//BsipZmYCh6gCX3f7q
IcjGX+R6HaQGeK7k97cNhX1WEooFPe/oajtPjou3td0jhf/9uJ6Jy6Ro=
```

**Note**  This is the signature that will be sent to the APIC for this specific request. Other requests will require to have their own signatures calculated.

Step 4  Place the signature inside a string to enable the APIC to verify the signature against the payload.

This complete signature is sent to the APIC as a cookie in the header of the request.

**Example:**

```
APIC-Request-Signature=P+OTqK0CeAZjl7+Gute2RlWw8OGgtzE0wsLlx8f
IXX14V79Zl7Ou8IdJH9CB4W6CEvdICxqkv3KaQazCIC0+Bn07o3qF//BsipZmYCh6gCX3f
IcjGX+R6HaQGeK7k97cNhX1WEooFPe/oajtPjou3td0jhf/9uJ6Jy6Ro=
```

**Note**  The DN used here must match the DN of the user certified object containing the x509 certificate in the next step.

Step 5  Use the CertSession class in the Python SDK to communicate with an APIC using signatures.

The following script is an example of how to use the CertSession class in the ACI Python SDK to make requests to an APIC using signatures.

**Example:**

```
#!/usr/bin/env python
# It is assumed the user has the X.509 certificate already added to
# their local user configuration on the APIC
from cobra.mit.session import CertSession
from cobra.mit.access import MoDirectory

def readFile(fileName=None, mode="r"):
    if fileName is None:
        return 
    fileData = 
    with open(fileName, mode) as aFile:
        fileData = aFile.read()
    return fileData

pkey = readFile("/tmp/userabc.key")
csession = CertSession("https://ApicIPOrHostname/",
    "uni/userext/user-userabc/usercert-userabc.crt", pkey)
modir = MoDirectory(csession)
resp = modir.lookupByDn('uni/fabric')
print resp.dn
```

**Note**  The DN used in the earlier step must match the DN of the user certified object containing the x509 certificate in this step.