Access, Authentication, and Accounting

- Overview, on page 1
- Configuration, on page 25
- Recovering Cisco APIC Passwords and Accessing Special Logins, on page 38

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.

Note There is a known limitation where you cannot have more than 32 characters for the login domain name. In addition, the combined number of characters for the login domain name and the user name cannot exceed 64 characters.

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 predefined or custom role, which is a set of one or more privileges assigned to a user
- A set of privileges, which determine the managed objects (MOs) to which the user has access

- 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

Roles and Privileges

A privilege controls access to a particular function within the system. The ACI fabric manages access privileges at the managed object (MO) level. Every object holds a list of the privileges that can read from it and a list of the privileges that can write to it. All objects that correspond to a particular function will have the privilege for that function in its read or write list. Because an object might correspond to additional functions, its lists might contain multiple privileges. When a user is assigned a role that contains a privilege, the user is given read access to the associated objects whose read list specifies read access, and write access to those whose write list specifies write access.

As an example, 'fabric-equipment' is a privilege that controls access to all objects that correspond to equipment in the physical fabric. An object corresponding to equipment in the physical fabric, such as 'eqptBoard,' will have 'fabric-equipment' in its list of privileges. The 'eqptBoard' object allows read-only access for the 'fabric-equipment' privilege. When a user is assigned a role such as 'fabric-admin' that contains the privilege 'fabric-equipment,' the user will have access to those equipment objects, including read-only access to the 'eqptBoard' object.

Note Some roles contain other roles. For example, '-admin' roles such as tenant-admin, fabric-admin, access-admin are groupings of roles with the same base name. For example, ‘access-admin’ is a grouping of 'access-connectivity', 'access-equipment', 'access-protocol', and 'access-qos.' Similarly, tenant-admin is a grouping of roles with a ‘tenant‘ base, and fabric-admin is a grouping of roles with a ‘fabric‘ base.

The 'admin' role contains all privileges.

For more details about roles and privileges see APIC Roles and Privileges Matrix.

Security Domains

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.

User Lockout After Continuous Failed Attempts to Log in

Starting in the 4.2(4) release, you can block a user from being able to log in after the user fails a configured number of login attempts. You can specify how many failed login attempts the user can have within a specific time period. If the user fails to log in too many times, then that user becomes unable to log in for a specified period of time.

This feature counts the failed login attempts both for local users that are in the Cisco Application Centric Infrastructure (ACI) database and for remote users who get authenticated with external authentication servers, such as RADIUS, LDAP, TACACS+, DUO Proxy, SAML, or RSA. A remote user who is locked out due to consecutive authentication failures using one external authentication server type will be locked out from all external authentication server types. For example, a user who is locked out after failing to log in using a RADIUS server will also be locked out when using an LDAP server. Authentications failing due to a AAA server being unreachable or down, or due to a bad SSH key, is not counted toward locking out a user; this feature only takes into account incorrect password entries.

A user who gets locked out from one Cisco Application Policy Infrastructure Controller (APIC) node in the cluster will be locked out from all nodes in the cluster, including the leaf switches and spine switches. A local user that does not exist in the Cisco ACI database cannot be locked out due to this feature.

Note You cannot configure this feature using the CLI.

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 With Cisco APIC release 5.0(1), the number of privileges was reduced from earlier releases, as many related legacy privileges were consolidated. Earlier privileges were remapped to current 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.

- Table 1: Privileges for Role: admin, on page 4
- Table 2: Privileges for Role: aaa, on page 5
- Table 3: Privileges for Role: access-admin, on page 5
- Table 4: Privileges for Role: fabric-admin, on page 5
- Table 5: Privileges for Role: nw-svc-admin, on page 6
- Table 6: Privileges for Role: nw-svc-params, on page 6
- Table 7: Privileges for Role: ops, on page 6
- Table 8: Privileges for Role: port-mgmt, on page 6
- Table 9: Privileges for Role: tenant-admin, on page 6
- Table 10: Privileges for Role: tenant-ext-admin, on page 8
- Table 11: Privileges for Role: vmm-admin, on page 9

Table 1: Privileges for Role: admin

Role: admin	
Privilege	**Description**
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.

Table 2: Privileges for Role: aaa

Role: aaa	
Privilege	**Description**
aaa	Used for configuring authentication, authorization, accounting, and import/export policies.

Table 3: Privileges for Role: access-admin

Role: access-admin	
Privilege	**Description**
access-connectivity	Used for Layer 1 to 3 configuration under infra, static route configurations under a tenant's L3Out, management infra policies, and tenant ERSPAN policies.
access-equipment	Used for access port configuration.
access-protocol	Used for Layer 1 to 3 protocol configurations under infra, fabric-wide policies for NTP, SNMP, DNS, and image management, and operations-related access policies such as cluster policy and firmware policies.
access-qos	Used for changing CoPP and QoS-related policies.

Table 4: Privileges for Role: fabric-admin

Role: fabric-admin	
Privilege	**Description**
fabric-connectivity	Used for Layer 1 to 3 configuration under the fabric, firmware and deployment policies for raising warnings for estimating policy deployment impact, and 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	Used for Layer 1 to 3 protocol configurations under the fabric, fabric-wide policies for NTP, SNMP, DNS, and image management, ERSPAN and health score policies, and firmware management traceroute and endpoint tracking policies.

Table 5: Privileges for Role: nw-svc-admin

Role: nw-svc-admin	
Privilege	**Description**
nw-svc-policy	Used for managing Layer 4 to Layer 7 service devices and network service orchestration.

Table 6: Privileges for Role: nw-svc-params

Role: nw-svc-params	
Privilege	**Description**
nw-svc-params	Used for managing Layer 4 to Layer 7 service policies.

Table 7: Privileges for Role: ops

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.

Table 8: Privileges for Role: port-mgmt

Role: port-mgmt	
Privilege	**Description**
port-mgmt	Used for assigning a node to a security domain. A user in a security domain with a Node Rule must also be assigned to domain `all` with the role of `port-mgmt`.

Table 9: Privileges for Role: tenant-admin

Role: tenant-admin	
Privilege	**Description**
aaa	Used for configuring authentication, authorization, accouting and import/export policies.
access-connectivity	Used for Layer 1 to 3 configuration under infra, static route configurations under a tenant's L3Out, management infra policies, and tenant ERSPAN policies.

Role: tenant-admin	
Privilege	**Description**
access-equipment	Used for access port configuration.
access-protocol	Used for Layer 1 to 3 protocol configurations under infra, fabric-wide policies for NTP, SNMP, DNS, and image management, and operations-related access policies such as cluster policy and firmware policies.
access-qos	Used for changing CoPP and QoS-related policies.
fabric-connectivity	Used for Layer 1 to 3 configuration under the fabric, firmware and deployment policies for raising warnings for estimating policy deployment impact, and 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	Used for Layer 1 to 3 protocol configurations under the fabric, fabric-wide policies for NTP, SNMP, DNS, and image management, ERSPAN and health score policies, and firmware management traceroute and endpoint tracking policies.
nw-svc-policy	Used for managing Layer 4 to Layer 7 service devices and 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.
tenant-connectivity	Used for Layer 1 to 3 connectivity changes, including bridge domains, subnets, and VRFs; for atomic counter, diagnostic, and image management policies on leaf switches and spine switches; tenant in-band and out-of-band management connectivity configurations; and 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	Used for write access firmware policies; managing tenant L2Out and L3Out configurations; and debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.

Role: tenant-admin	
Privilege	**Description**
tenant-ext-protocol	Used for managing tenant external Layer 1 to 3 protocols, including BGP, OSPF, PIM, and IGMP, and for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk. Generally only used for write access for firmware policies.
tenant-network-profile	Used for managing tenant configurations, such as deleting and creating network profiles, and deleting and creating endpoint groups.
tenant-protocol	Used for managing configurations for Layer 1 to 3 protocols under a tenant, for tenant traceroute policies, and as write access for firmware policies.
tenant-qos	Used for QoS-related configurations for a tenant.
tenant-security	Used for contract-related configurations for a tenant.
vmm-policy	Used for managing policies for virtual machine networking, such as authentication and connectivity.

Table 10: Privileges for Role: tenant-ext-admin

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

Role: tenant-ext-admin	
Privilege	**Description**
tenant-qos	Used for QoS-related configurations for a tenant.
tenant-security	Used for contract-related configurations for a tenant.
vmm-policy	Used for managing policies for virtual machine networking, such as authentication and connectivity.

Table 11: Privileges for Role: vmm-admin

Role: vmm-admin	
Privilege	**Description**
vmm-policy	Used for managing policies for virtual machine networking, such as authentication and connectivity.

Remapping of Legacy Privileges

With Cisco APIC Release 5.0(1), the number of privileges is reduced from earlier releases, as many related legacy privileges are consolidated. The following tables show the mapping of earlier privileges to the current privileges.

- Table 12: Privilege: admin, on page 10
- Table 13: Privilege: aaa, on page 10
- Table 14: Privilege: access-connectivity, on page 10
- Table 15: Privilege: access-equipment, on page 10
- Table 16: Privilege: access-protocol, on page 11
- Table 17: Privilege: access-qos, on page 11
- Table 18: Privilege: fabric-connectivity, on page 11
- Table 19: Privilege: fabric-equipment, on page 11
- Table 20: Privilege: fabric-protocol, on page 12
- Table 21: Privilege: nw-svc-params, on page 12
- Table 22: Privilege: nw-svc-policy, on page 12
- Table 23: Privilege: ops, on page 12
- Table 24: Privilege: tenant-connectivity, on page 12
- Table 25: Privilege: tenant-epg, on page 13
- Table 26: Privilege: tenant-ext-connectivity, on page 13
- Table 27: Privilege: tenant-ext-protocol, on page 13

- Table 28: Privilege: tenant-network-profile, on page 13
- Table 29: Privilege: tenant-protocol, on page 14
- Table 30: Privilege: tenant-qos, on page 14
- Table 31: Privilege: tenant-security, on page 14
- Table 32: Privilege: vmm-policy, on page 14

Table 12: Privilege: admin

Previous Privileges	Current Privilege
admin	admin

Table 13: Privilege: aaa

Previous Privileges	Current Privilege
aaa	aaa

Table 14: Privilege: access-connectivity

Previous Privileges	Current Privilege
access-connectivity-l1	access-connectivity
access-connectivity-l2	
access-connectivity-l3	
access-connectivity-mgmt	
access-connectivity-util	

Table 15: Privilege: access-equipment

Previous Privileges	Current Privilege
access-equipment	access-equipment

Table 16: Privilege: access-protocol

Previous Privileges	Current Privilege
access-protocol-l1	access-protocol
access-protocol-l2	
access-protocol-l3	
access-protocol-mgmt	
access-protocol-ops	
access-protocol-util	

Table 17: Privilege: access-qos

Previous Privileges	Current Privilege
access-qos	access-qos

Table 18: Privilege: fabric-connectivity

Previous Privileges	Current Privilege
fabric-connectivity-l1	fabric-connectivity
fabric-connectivity-l2	
fabric-connectivity-l3	
fabric-connectivity-mgmt	
fabric-connectivity-util	

Table 19: Privilege: fabric-equipment

Previous Privileges	Current Privilege
fabric-equipment	fabric-equipment

Table 20: Privilege: fabric-protocol

Previous Privileges	Current Privilege
fabric-protocol-l1	fabric-protocol
fabric-protocol-l2	
fabric-protocol-l3	
fabric-protocol-mgmt	
fabric-protocol-ops	
fabric-protocol-util	

Table 21: Privilege: nw-svc-params

Previous Privileges	Current Privilege
nw-svc-params	nw-svc-params

Table 22: Privilege: nw-svc-policy

Previous Privileges	Current Privilege
nw-svc-devshare	nw-svc-policy
nw-svc-params	
nw-svc-policy	

Table 23: Privilege: ops

Previous Privileges	Current Privilege
ops	ops

Table 24: Privilege: tenant-connectivity

Previous Privileges	Current Privilege
tenant-connectivity-util	tenant-connectivity
tenant-connectivity-l2	
tenant-connectivity-l3	
tenant-connectivity-mgmt	
tenant-connectivity-util	

Table 25: Privilege: tenant-epg

Previous Privileges	Current Privilege
tenant-epg	tenant-epg

Table 26: Privilege: tenant-ext-connectivity

Previous Privileges	Current Privilege
tenant-ext-connectivity-l1	tenant-ext-connectivity
tenant-ext-connectivity-l2	
tenant-ext-connectivity-l3	
tenant-ext-connectivity-mgmt	
tenant-ext-connectivity-util	

Table 27: Privilege: tenant-ext-protocol

Previous Privileges	Current Privilege
tenant-ext-protocol-l1	tenant-ext-protocol
tenant-ext-protocol-l2	
tenant-ext-protocol-l3	
tenant-ext-protocol-mgmt	
tenant-ext-protocol-util	

Table 28: Privilege: tenant-network-profile

Previous Privileges	Current Privilege
tenant-network-profile	tenant-network-profile

Table 29: Privilege: tenant-protocol

Previous Privileges	Current Privilege
tenant-protocol-l1	tenant-protocol
tenant-protocol-l2	
tenant-protocol-l3	
tenant-protocol-mgmt	
tenant-protocol-ops	
tenant-protocol-util	

Table 30: Privilege: tenant-qos

Previous Privileges	Current Privilege
tenant-qos	tenant-qos

Table 31: Privilege: tenant-security

Previous Privileges	Current Privilege
tenant-security	tenant-security

Table 32: Privilege: vmm-policy

Previous Privileges	Current Privilege
vmm-connectivity	vmm-policy
vmm-ep	
vmm-policy	
vmm-protocol-ops	
vmm-security	

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	

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:

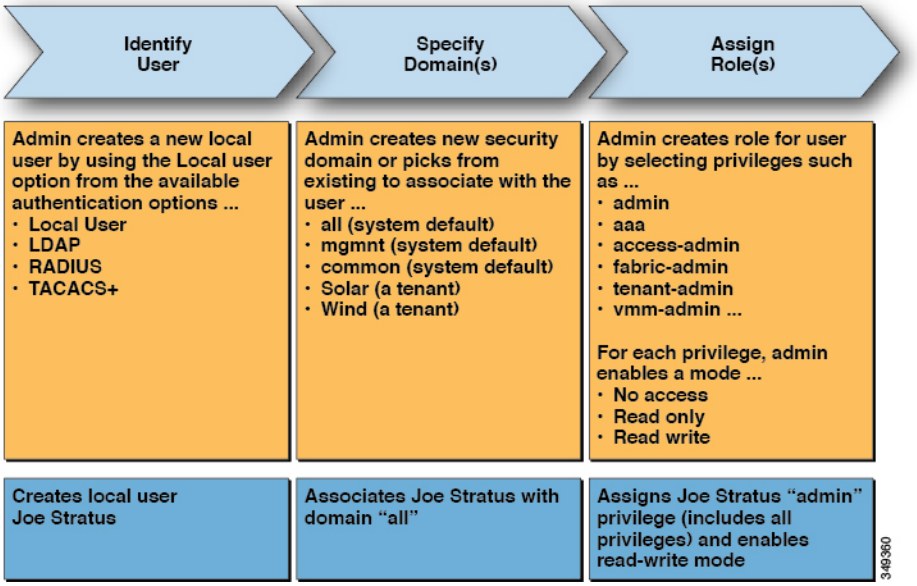
- Minimum password length is 8 characters.
- Maximum password length is 64 characters.
- Has fewer than three consecutive repeated characters.
- Must have characters from at least three of the following characters types: lowercase, uppercase, digit, symbol.
- Does not use easily guessed passwords.
- 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.

Cisco ACI uses a crypt library with a SHA256 one-way hash for storing passwords. At rest hashed passwords are stored in an encrypted filesystem. The key for the encrypted filesystem is protected using the Trusted Platform Module (TPM).

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.

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.

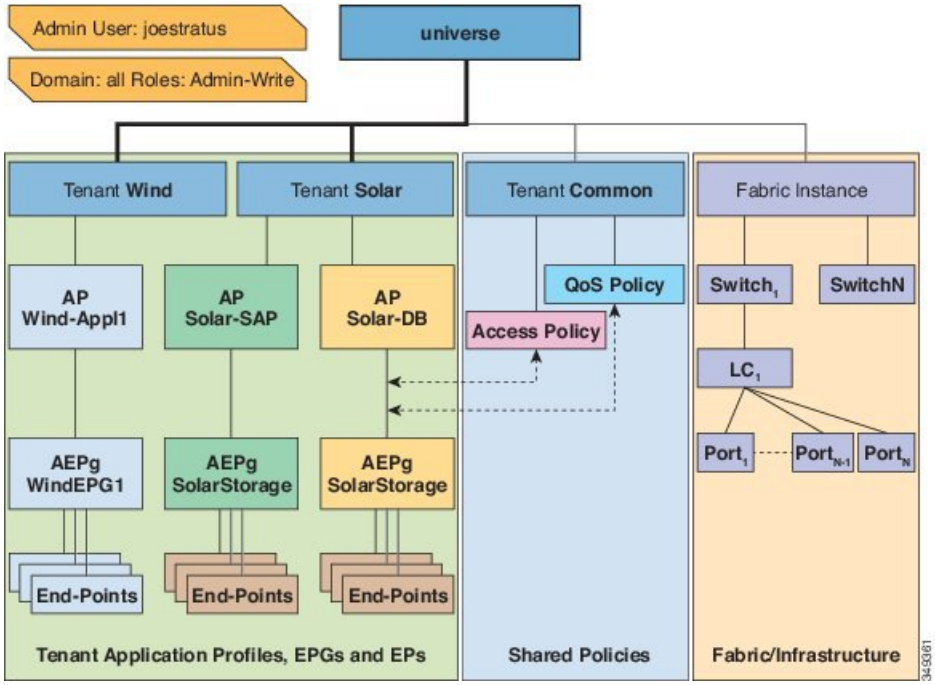
Figure 1: APIC Local User Configuration Process



Note 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.

OTP-Based Two-Factor Authentication for Local Users

A fabric admin user can enable the one-time password (OTP) feature for a local user. A one-time password changes every 30 seconds for enhanced security. After the admin enables OTP, Cisco Application Policy Infrastructure Controller (APIC) generates a random human-readable 16 binary octet that is a base32 OTP key. This OTP key is used to generate the OTP for the user, which is used for two-factor authentication.

Cisco APIC supports the following security platforms for use with two-factor authentication:

- Duo Security, with the Duo Mobile app
- Google, with the Google Authenticator app (only with Android and Apple iOS smartphones)

Note You must download the indicated app from the appropriate app store.

These security platforms do not act as a repository for user identities. The platforms offer two-factor authentication on top of an organization's existing authentication, which could be on-premises or cloud-based. Two-factor authentication occurs once the user has finished the authentication with the organization's primary authentication source.

The platforms support three types of two-factor authentication methods after you complete authentication with the primary authentication source:

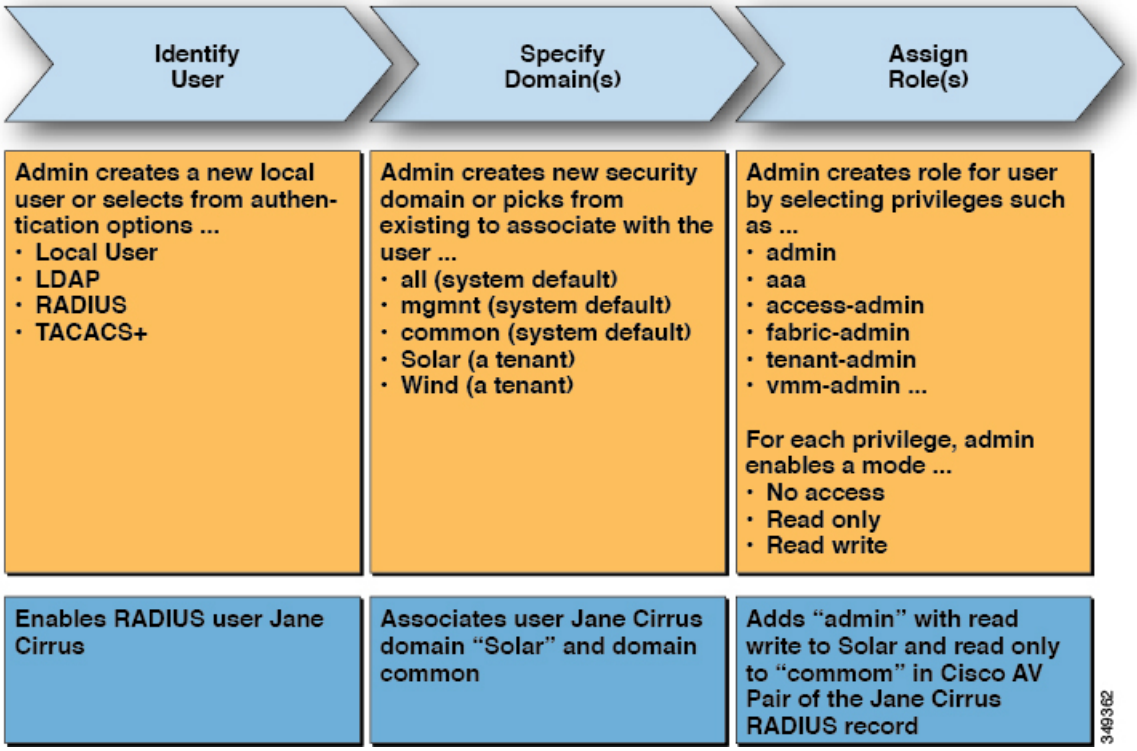
- Notification push on mobile using the appropriate mobile app on smartphones.

- Phone call on your registered phone or mobile numbers.
- Passcode that is generated on the appropriate mobile app.

Externally Managed Authentication Server Users

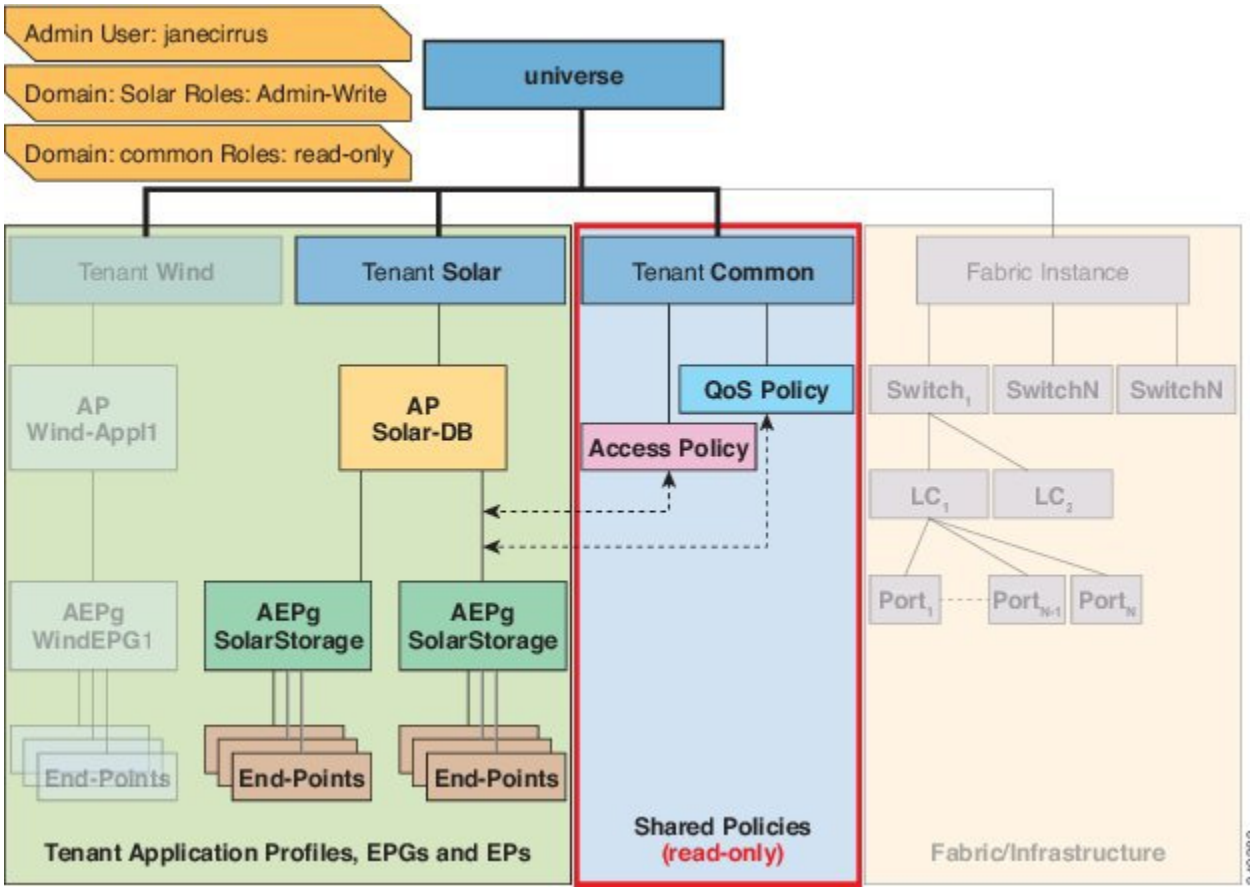
The following figure shows how the process works for configuring an admin user in an external RADIUS server who has full access to the tenant Solar.

Figure 3: Process for Configuring Users on External Authentication Servers



The following figure shows the access the admin user Jane Cirrus has to the system.

Figure 4: Result of Configuring Admin User for Tenant Solar



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

- `common` is the tenant-common subtree that all users should have read-only access to
- `read-all` is the role with read 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+?/\\S*?/\\S*?){0,31})(\\(\\d+\\))$
shell:domains\\s*[=:]\\s*((\\S+?/\\S*?/\\S*?)(,\\S+?/\\S*?/\\S*?){0,31})$
```

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
```

Note 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:

Note
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:

```
<!--   api/requestNewRole/json   -->
<aaaChangeRole>
<attributes userName="apic#tacacs” apicadmin="pwd Ins3965!” role=“newrole"/>
```

Step 2 Return to the original role:

Example:

```
<!--   api/requestNewRole/json   -->
<aaaChangeRole>
<attributes userName="apic#tacacs” apicadmin="pwd Ins3965!” role=“"/>
```

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.

Note 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

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

- The `aaaSessionLR` managed object tracks user account login and logout sessions on the Cisco Application Policy Infrastructure Controller (APIC) and switches, and token refresh. The Cisco ACI fabric session alert feature stores information such as the following:
  - Username
  - IP address initiating the session
  - Type (telnet, HTTPS, REST, and so on)

- 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 Cisco ACI fabric.

Note 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` managed object 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 Cisco APIC shards. After the data exceeds the pre-set storage allocation size, it overwrites records on a first-in first-out basis.

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

The `aaaModLR` and `aaaSessionLR` managed objects 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 Cisco 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` managed object query data. There is no default policy to export this data.

There are no pre-configured queries in the Cisco 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 Cisco Application Policy Infrastructure Controller (APIC) can be configured to collect byte count and packet count billing statistics from a port configured for routed connectivity to external networks as a shared service. The external networks are represented as external L3Out endpoint group (l3extInstP managed object) in Cisco Application Centric Infrastructure (ACI). Any EPG in any tenant can share an external L3Out EPG for routed connectivity to external networks. Billing statistics can be collected for each EPG in any tenant that uses an external L3Out EPG as a shared service. The leaf switch where the external L3Out EPG is provisioned forwards the billing statistics to the Cisco APIC where they are aggregated. Accounting policies can be configured to export these billing statics periodically 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 user 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+ provider.
  - 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

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

Step 2 In the **Navigation** pane, click **Users**.

In the **Work** pane, verify that you are in the **Local Users** tab.

Step 3 In the **Work** pane, click the task icon drop-down list and select **Create Local User**.

Step 4 In the **STEP 1 > User Identity** dialog box, perform the following actions:

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

The login ID must meet the following guidelines:

- Must be unique within APIC.
- Must begin with a letter.
- Can contain between 1 and 32 characters.
- Can include alphanumeric characters, underscores, dashes, and dots.

After creating a user account, you cannot change the login ID. You must delete the user account and create a new one.

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.
- Cannot be any variation of cisco, isco or any permutation of these characters or variants obtained by changing the capitalization of letters therein.

c) In the **Confirm Password** field, confirm the password.
d) (Optional) For Certificate based authentication, in the **User Certificate Attribute** field, enter the user identity from the authentication certificate.
e) Click **Next**.

Step 5 You can activate or deactivate the user account by using the **Account Status** control, and you can set an expiration date by using the **Account Expires** control.

Step 6 In the **STEP 2 > Security** dialog box, under **Security Domain**, choose the desired security domain for the user, and click **Next**.

Step 7 In the **STEP 3 > Roles** dialog box, perform the following actions:

a) Click the + to associate the user with a domain.
b) From the drop-down lists, choose a **Role Name** and a **Role Privilege Type** for the user.
c) click **Update**

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

Step 8 click **Finish**

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** > **Users** and confirm you are in the **Local Users** tab.

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

SUMMARY STEPS

1. In the NX-OS CLI, start in configuration mode, shown as follows:
2. Create a new user, shown as follows:

DETAILED STEPS

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:

```
apic1(config)# username
 WORD       User name (Max Size 28)
 admin
 cli-user
 jigarshah
 test1
 testUser

apic1(config)# username test
apic1(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

apic1(config-username)# exit
```

Configuring a Local User Using the REST API

SUMMARY STEPS

1. Create a local user.

DETAILED STEPS

Procedure

Create a local user.

Example:

```
URL: https://apic-ip-address/api/node/mo/uni/userext.xml
POST CONTENT:
    <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)
                Serial Number:
                    c4:27:6c:4d:69:7c:d2:b6
                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: CN=User ABC, O=Cisco Systems, C=US
                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:
                            99:d3:25:42:94:b5:3e:8a:32:55:ce:e9:21:2a:ff:
                            e0:e4:22:58:6d:40:98:b1:0d:42:21:db:cd:44:26:
                            50:77:e5:fa:b6:10:57:d1:ec:95:e9:86:d7:3c:99:
                            ce:c4:7f:61:1d:3c:9e:ae:d8:88:be:80:a0:4a:90:
                            d2:22:e9:1b:25:27:cd:7d:f3:a5:8f:cf:16:a8:e1:
                            3a:3f:68:0b:9c:7c:cb:70:b9:c7:3f:e8:db:85:d8:
                            98:f6:e3:70:4e:47:e2:59:03:49:01:83:8e:50:4a:
                            5f:bc:35:d2:b1:07:be:ec:e1
                        Exponent: 65537 (0x10001)
                X509v3 extensions:
                    X509v3 Subject Key Identifier:
                        0B:E4:11:C7:23:46:10:4F:D1:10:4C:C1:58:C2:1E:18:E8:6D:85:34
                    X509v3 Authority Key Identifier:
                        keyid:0B:E4:11:C7:23:46:10:4F:D1:10:4C:C1:58:C2:1E:18:E8:6D:85:34
                        DirName:/CN=User ABC/O=Cisco Systems/C=US
                        serial:C4:27:6C:4D:69:7C:D2:B6

                    X509v3 Basic Constraints:
                        CA:TRUE
            Signature Algorithm: sha1WithRSAEncryption
                8f:c4:9f:84:06:30:59:0c:d2:8a:09:96:a2:69:3d:cf:ef:79:
                91:ea:cd:ae:80:16:df:16:31:3b:69:89:f7:5a:24:1f:fd:9f:
                d1:d9:b2:02:41:01:b9:e9:8d:da:a8:4c:1e:e5:9b:3e:1d:65:
                84:ff:e8:ad:55:3e:90:a0:a2:fb:3e:3e:ef:c2:11:3d:1b:e6:
                f4:5e:d2:92:e8:24:61:43:59:ec:ea:d2:bb:c9:9a:7a:04:91:
                8e:91:bb:9d:33:d4:28:b5:13:ce:dc:fe:c3:e5:33:97:5d:37:
                cc:5f:ad:af:5a:aa:f4:a3:a8:50:66:7d:f4:fb:78:72:9d:56:
                91:2c
        [snip]
```

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 **Users** and **Local Users** in the **Work** pane.

Step 3 In the **Work** pane, verify that you in the **Local Users** tab.

The admin user is present by default

Step 4 In the **Work** pane, click on task icon drop-down list and select **Create Local User**.

Step 5 In the **Security** dialog box, choose the desired security domain for the user, and click **Next**.

Step 6 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 7 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) (Optional) For Certificate based authentication, in the **User Certificate Attribute** field, enter the user identity from the authentication certificate.
e) Click **Finish**.

Step 8 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 9 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:
        {
            "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-----\nMIICjjCCAfegAwIBAgIJAMQnbE <snipped
content> ==\n-----END CERTIFICATE-----",
                        },
                        "children": []
                    },
                    "aaaUserDomain": {
                        "attributes": {
                            "name": "all",
                        },
                        "children": [{
                            "aaaUserRole": {
                                "attributes": {
                                    "name": "aaa",
                                    "privType": "writePriv",
                                },
                                "children": []
                            }
                        }, {
                            "aaaUserRole": {
                                "attributes": {
                                    "name": "access-admin",
                                    "privType": "writePriv",
                                },
                                "children": []
                            }
                        }, {
                            "aaaUserRole": {
                                "attributes": {
                                    "name": "admin",
                                    "privType": "writePriv",
                                },
                                "children": []
                            }
                        }, {
                            "aaaUserRole": {
                                "attributes": {
                                    "name": "fabric-admin",
                                    "privType": "writePriv",
                                },
                                "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:

```
#!/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)

cr = ConfigRequest()
cr.addMo(aaaUser)
modir.commit(cr)
```

```
# 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 Verify that the payload.txt file contains the correct information.

For example, using the GET example shown in the previous step:

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

Your payload.txt file should contain only the following information:

```
GET/api/class/fvTenant.json?rsp-subtree=children
```

Step 3 Verify that you didn't inadvertently create a new line when you created the payload file.

Example:

```
# cat -e payload.txt
```

Determine if there is a **$** symbol at the end of the output, similar to the following:

```
GET/api/class/fvTenant.json?rsp=subtree=children$
```

If so, then that means that a new line was created when you created the payload file. To prevent creating a new line when generating the payload file, use a command similar to the following:

```
echo -n "GET/api/class/fvTenant.json?rsp-subtree=children" >payload.txt
```

Step 4 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 5 Convert the signature to base64 format:

Example:

```
openssl base64 -A -in payload_sig.bin -out payload_sig.base64
```

Step 6 Strip the signature of the new lines using Bash.

Example:

```
$ tr -d '\n' < payload_sig.base64
P+OTqK0CeAZjl7+Gute2R1Ww8OGgtzE0wsLlx8fIXXl4V79Zl7
Ou8IdJH9CB4W6CEvdICXqkv3KaQszCIC0+Bn07o3qF//BsIplZmYChD6gCX3f7q
IcjGX+R6HAqGeK7k97cNhXlWEoobFPe/oajtPjOu3tdOjhf/9ujG6Jv6Ro=
```

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 7 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+Gute2R1Ww8OGgtzE0wsLlx8f
IXXl4V79Zl7Ou8IdJH9CB4W6CEvdICXqkv3KaQszCIC0+Bn07o3qF//BsIplZmYChD6gCX3f
7qIcjGX+R6HAqGeK7k97cNhXlWEoobFPe/oajtPjOu3tdOjhf/9ujG6Jv6Ro=;
APIC-Certificate-Algorithm=v1.0; APIC-Certificate-Fingerprint=fingerprint;
APIC-Certificate-DN=uni/userext/user-userabc/usercert-userabc.crt
```

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

Step 8 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", pkey)

modir = MoDirectory(csession)
resp = modir.lookupByDn('uni/fabric')
pring resp.dn
# End of script
```

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

Configuring User Lockout After Continuous Failed Attempts to Log in using the GUI

You can block a user from being able to log in after the user fails a configured number of login attempts. You can specify how many failed login attempts the user can have within a specific time period. If the user fails to log in too many times, then that user becomes unable to log in for a specified period of time.

Procedure

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

Step 2 In the **Navigation** pane, choose **Security**.

Step 3 In the **Work** pane, choose the **Management Settings** > **Policy** tab, if it is not chosen already.

Step 4 Under **Properties**, fill out the fields as follows:

a) For **Lockout User after multiple failed login attempts**, choose **Enable**.

b) For **Number of failed attempts before user is locked out**, enter the desired value.

The range is from 1 to 15. The default is 5.

c) For **Time period in which consecutive attempts were failed (m)**, enter a value in minutes for the time interval during which the Cisco Application Policy Infrastructure Controller (APIC) will count the failed attempts.

The range is from 1 to 720. The default is 5.

d) For **Duration of lockout (m)**, enter a value in minutes for how long a user will be locked out for failing to log in too many times.

Step 5 Click **Submit**.

Configuring a Local User for OTP-Based Two-Factor Authentication Using the GUI

The following procedure configures OTP-based two-factor authentication for a local user using the Cisco APIC GUI. The procedure assumes that you are a fabric administrator.

Before you begin

You must have already created a local user for which you want to enable OTP-based two-factor authentication.

Procedure

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

Step 2 In the **Navigation** pane, choose **Users**.

Step 3 Double-click the user for which you want to enable OTP-based two-factor authentication.

Step 4 For **Enable OTP**, choose **Yes**.

Step 5 Click **Submit**.

Step 6 In the **Policy Usage Warning** dialog, click **Submit Changes**.

From this point, you can provide the QR code or OTP key to the user such as through an email, or the user can log into the Cisco APIC GUI to get the code and key directly.

Step 7 (Optional) If you want to provide the QR code or OTP key to the user, continue with the substeps. Otherwise, stop here.

a) Double-click the user again.

b) To the right of **OTP Key**, click the expand button.

A dialog displays that includes the QR code and OTP key.

c) Provide the QR code or OTP key to the user.

d) In the Cisco APIC, click **Close**.

e) Click **Close**.

What to do next

The user for which you enabled OTP must complete the configuration of OTP authentication. See Completing the Configuration of OTP-Based Two-Factor Authentication by a User Using the GUI, on page 37.

Completing the Configuration of OTP-Based Two-Factor Authentication by a User Using the GUI

The following procedure completes the configuration of OTP-based two-factor authentication using the Cisco APIC GUI. The procedure assumes that you are a user for which a fabric administrator enabled OTP-based two-factor authentication.

Before you begin

A fabric administrator must have enabled OTP-based two-factor authentication for your account.

Procedure

Step 1 On your Android or Apple iOS smartphone, download the appropriate two-factor authentication app.

Step 2 Get the QR code or OTP key from the fabric administrator or by logging in to the Cisco APIC GUI.

If you log into the GUI, the QR code and OTP key display after you enter your credentials.

Step 3 Using your smartphone, scan the QR code and follow the two-factor authentication app's directions, or enter the OTP key in the Cisco APIC GUI.

Recovering Cisco APIC Passwords and Accessing Special Logins

Recovering the APIC Password

For Cisco APIC Release 5.3 or 6.0 or later, contact Cisco TAC to recover the Cisco Application Policy Infrastructure Controller (APIC) password. You cannot recover the password on your own.

Follow these steps to recover the APIC password.

Procedure

Step 1 Create and save an empty file named "aci-admin-passwd-reset.txt".

Step 2 Add the file to a USB drive. You can format the USB drive to FAT or FAT32.

Step 3 Connect the USB drive to one of the rear USB ports on the Cisco APIC.

Step 4 Reboot the Cisco APIC using Cisco Integrated Management Controller (CIMC) or by hard power cycling the device.

Step 5 Press the **Esc** key during the 10-second countdown timer that appears at the top left to bring up the list of boot targets.

Step 6 Press the **e** key to edit the default grub line.

Step 7 Go to the line that begins with "linux." Using the **End** key or **Right Arrow** key, move the cursor to the end of that line and append "aci-admin-passwd-reset".

Step 8 Press **Ctrl+X** to boot the entry.

It may take a few minutes for the new password to take effect.

Using the Rescue-user Account to Erase the Cisco APIC Configuration Using the NX-OS Style CLI

The rescue-user is an emergency login that provides access to the Cisco APIC even when it is not in a cluster. You can use this login to run troubleshooting commands including erasing the configuration.

Note For a standby Cisco APIC, you can log in using SSH with the username "rescue-user" and no password. If the standby Cisco APIC was previously part of a fabric, the "rescue-user" account will retain the old administrator password, unless the operating system is re-installed using the keyboard, video, mouse (KVM) console.

Procedure

Step 1 Access the APIC using the Cisco Integrated Management Controller (CIMC) console.

Step 2 Login as rescue-user.

Note
If an admin password is in place and the Cisco APIC is logged onto the fabric, the rescue-user password is the same as the admin password. Otherwise there is no rescue-user password.

Step 3 Use the **acidiag touch** command to clear the configuration.

Example:

```
apic1# acidiag touch setup
```

Using the Fallback Login Domain to Log in to the Local Database

There is a hidden login domain named "fallback" that allows you to log in using the local user database in case of lockout. The format of the username used for the authentication method is `apic#fallback\\<username>`.

Procedure

Step 1 Use the fallback login domain to log in to the local database in the GUI or log in to the fallback login domain using the NX-OS-style CLI, shown as follows:

```
apic1(config)# aaa authentication login domain fallback
apic1(config-domain)# ?
group Set provider group for login domain
realm Specify server realm
```

Step 2 Optionally, you can instead use the REST API to log in to the fallback login domain, shown as follows:

- URL: `https://ip_address/api/aaaLogin.xml`
- DATA:

```
<aaaUser name="apic#fallback\\admin"
pwd="passwordhere"/>
```