User Access, Authentication, and Accounting

This chapter contains the following sections:

Access Rights Workflow Dependencies

The Cisco 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.

User Access, Authentication, and Accounting

APIC policies manage the access, authentication, and accounting (AAA) functions of the Cisco 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 very granular fashion. These configurations can be implemented using the REST API, the CLI, or the GUI.

Multiple Tenant Support

A core 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 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 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.

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 greyed out.

A set of pre-defined 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.

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

Note To watch an example video of this task, see Videos Webpage.

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

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 **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.
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) 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.

Configuring a Local User Using the NX-OS Style CLI

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 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:

```
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 Remote User

Instead of configuring local users, you can point the APIC at the centralized enterprise credential datacenter. The APIC supports Lightweight Directory Access Protocol (LDAP), active directory, RADIUS, and TACACS+.

To configure a remote user authenticated through an external authentication provider, you must meet the following prerequisites:

- The DNS configuration should have already been resolved with the hostname of the RADIUS server.

- You must configure the management subnet.

AV Pair on the External Authentication Server

You can add a Cisco attribute/value (AV) pair to the existing user record to propagate the user privileges to the APIC controller. The Cisco AV pair is a single string that you use to specify the Role-Based Access Control (RBAC) roles and privileges for an APIC user. An example configuration for an open RADIUS server (/etc/raddb/users) is as follows:

```
aaa-network-admin Cleartext-Password := "<password>"
Cisco-avpair = "shell:domains = all/aaa/read-all(16001)"
```

Best Practice for Assigning AV Pairs

As best practice, Cisco recommends that you assign unique UNIX user ids in the range 16000-23999 for the AV Pairs that are assigned to users when in bash shell (using SSH, Telnet or Serial/KVM consoles). If a situation arises when the Cisco AV Pair does not provide a UNIX user id, the user is assigned a user id of 23999 or similar number from the range that also enables the user's home directories, files, and processes accessible to remote users with a UNIX ID of 23999.

Configuring an AV Pair on the External Authentication Server

The numerical value within the parentheses in the attribute/value (AV) pair string is used as the UNIX user ID of the user who is logged in using Secure Shell (SSH) or Telnet.

Procedure

Configure an AV pair on the external authentication server.
The Cisco AV pair definition is as follows (Cisco supports AV pairs with and without UNIX user IDs specified):

Example:

```
* shell:domains =
domainA/writeRole1|writeRole2|writeRole3/readRole1|readRole2,domainB/writeRole1|writeRole2|writeRole3/readRole1|readRole2

         * shell:domains =
domainA/writeRole1|writeRole2|writeRole3/readRole1|readRole2,domainB/writeRole1|writeRole2|writeRole3/readRole1|readRole2(8101)

These are the boost regexes supported by APIC:
uid_regex("shell:domains\\s*[=:]\\s*((\\S+?/\\S*?/\\S*?)(,\\S+?/\\S*?/\\S*?){0,31})(\\(\\d+\\))$");
regex("shell:domains\\s*[=:]\\s*((\\S+?/\\S*?/\\S*?)(,\\S+?/\\S*?/\\S*?){0,31})$");
```

The following is an example:

```
shell:domains = coke/tenant-admin/read-all,pepsi//read-all(16001)
```

Configuring APIC for TACACS+ Access

Before You Begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- The TACACS+ server host name or IP address, port, and key are available.
- The APIC management EPG is available.

- An APIC user account is available that will enable the following:
  - Creating the TACACS+ provider and TACACS+ provider group.

Figure 1: TACACS+ Provider

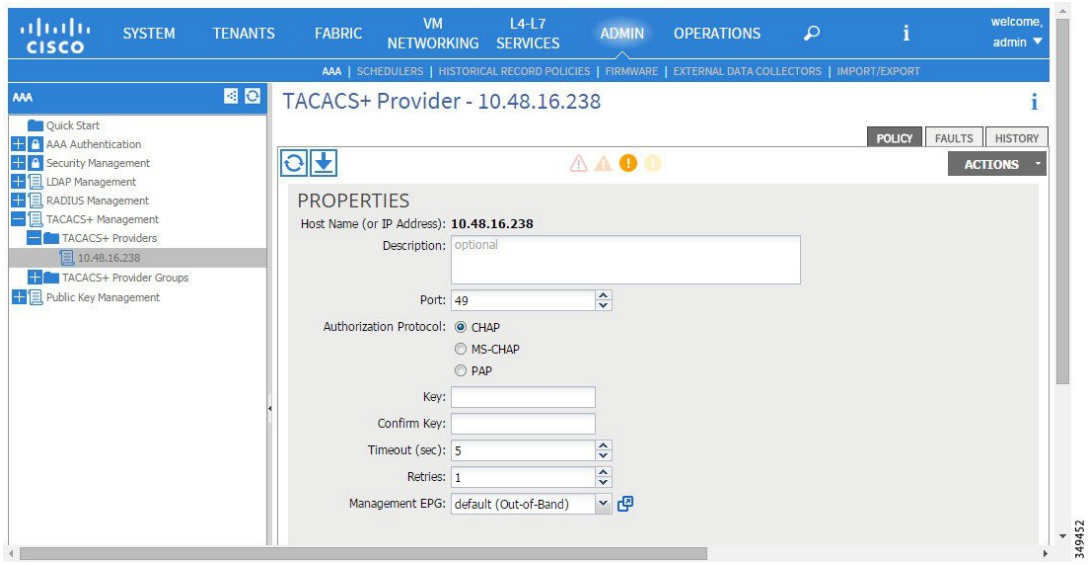
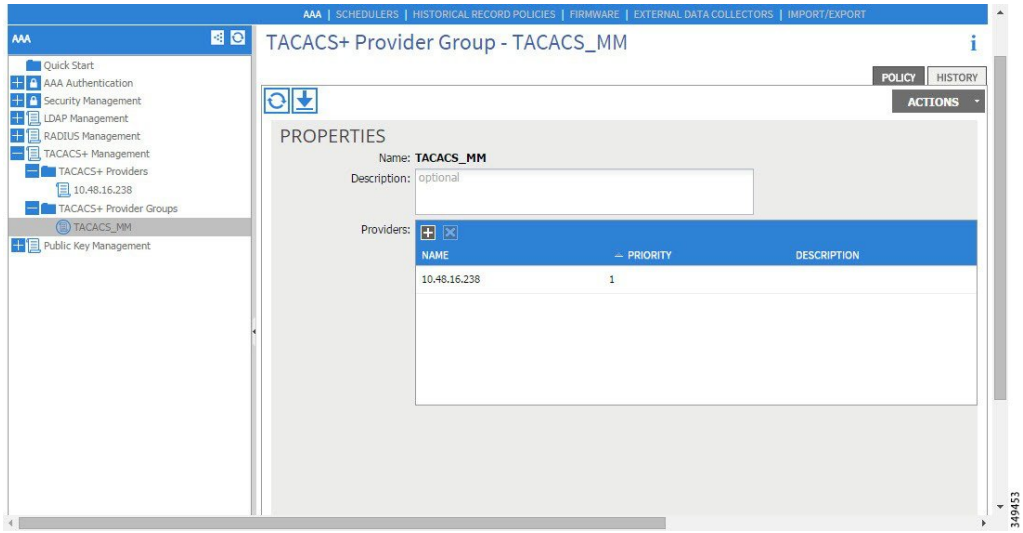
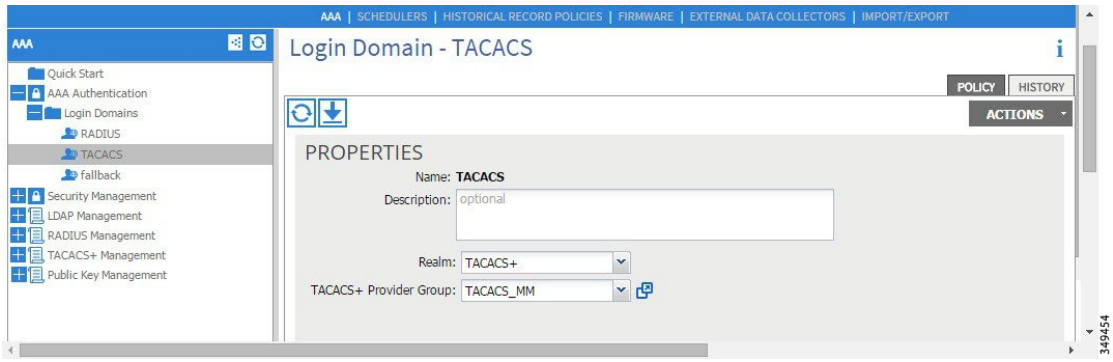


Figure 2: TACACS+ Provider Group



- Creating the TACACS+ login domain.

Figure 3: AAA Login Domain for TACACS+



Procedure

Step 1 On the APIC, create the **TACACS+ Provider**.

a) On the APIC menu bar, click **ADMIN** > **AAA**.

b) In the **Navigation** pane, click the + icon to expand the**TACACS+ Managment** option.

c) In the **Navigation** pane, right-click the **TACACS+ Providers** option, and select **Create TACACS+ Provider**.

d) Specify the TACACS+ host name (or IP address), port, authorization protocol, key, and management EPG.

Note If the APIC is configured for in-band management connectivity, choosing an out-of-band management EPG for TACACS+ access does not take effect. Alternatively, an out-of-band over an in-band management EPG can connect a TACACS+ server but requires configuring a static route for the TACACS+ server. The Cisco ACS sample configuration procedure below uses an APIC in-band IP address.

Step 2 Create the **TACACS+ Provider Group**.

a) In the **Navigation** pane, right-click the the**TACACS+ Provider Groups** option, and select **Create TACACS+ Provider Group**

b) Specify the TACACS+ Provider Group name, description, and provider(s) as appropriate.

Step 3 Create the **Login Domain** for TACACS+.

a) In the **Navigation** pane, click the + icon to expand the**AAA Authentication** option.

b) In the **Navigation** pane, right-click the **Login Domains** option, and select **Create Login Domain**.

c) Specify the Login Domain name, description, realm, and provider group as appropriate.

What to Do Next

This completes the APIC TACACS+ configuration steps. Next, if a RAIDUS server will also be used, configure the APIC for RADIUS. If only a TACACS+ server will be used, go to the ACS server configuration topic below.

Configuring APIC for RADIUS Access

Before You Begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- The RADIUS server host name or IP address, port, authorization protocol, and key are available.
- The APIC management EPG is available.
- An APIC user account is available that will enable the following:

- Creating the RADIUS provider and RADIUS provider group.

Figure 4: RADIUS Provider

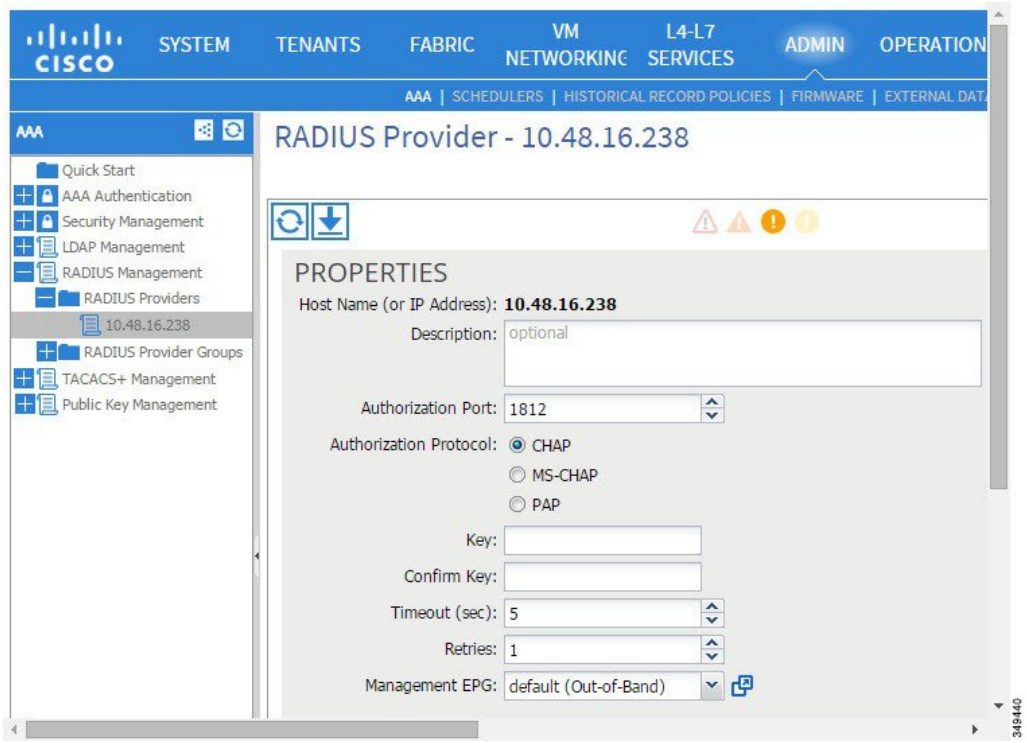
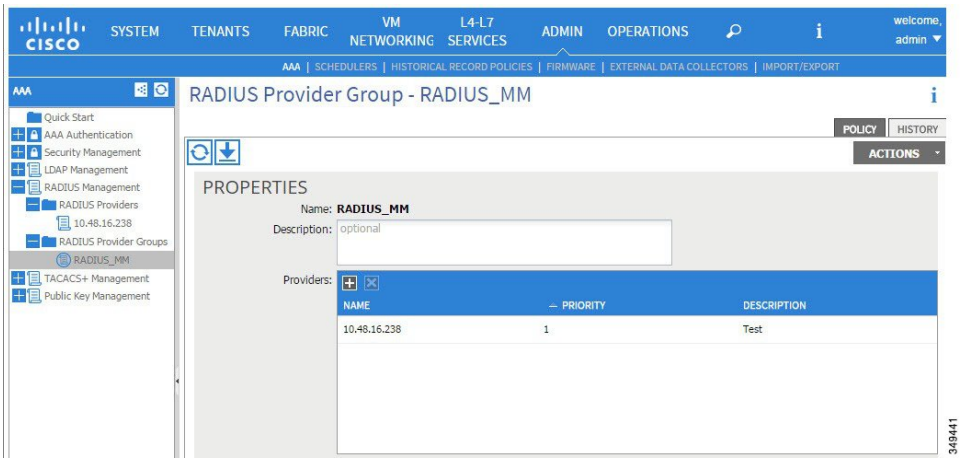
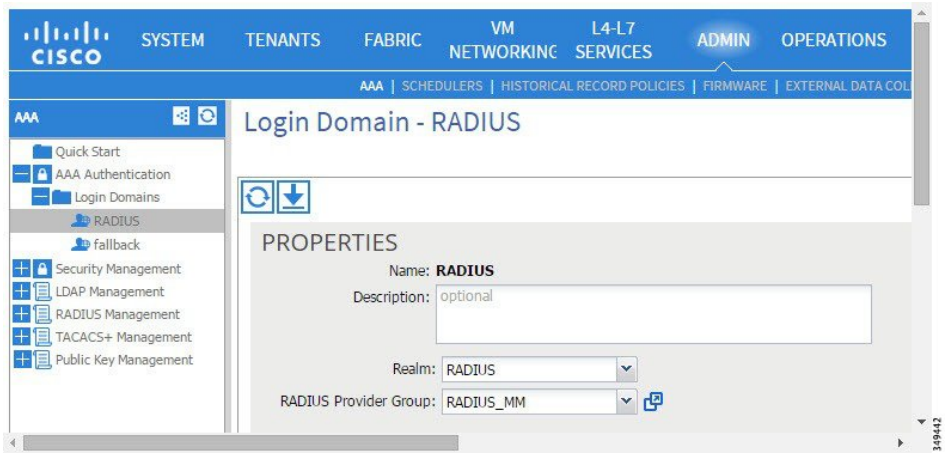


Figure 5: RADIUS Provider Group



- Creating the RADIUS login domain.

Figure 6: AAA Login Domain for RADIUS



Procedure

Step 1 On the APIC, create the **RADIUS Provider**.

a) On the APIC menu bar, click **ADMIN** > **AAA**.

b) In the **Navigation** pane, click the + icon to expand the**RADIUS Managment** option.

c) In the **Navigation** pane, right-click the **RADIUS Providers** option, and select **Create RADIUS Provider**.

d) Specify the RADIUS host name (or IP address), port, protocol, and management EPG.

Note If the APIC is configured for in-band management connectivity, choosing an out-of-band management EPG for RADIUS access does not take effect. Alternatively, an out-of-band over an in-band management EPG can connect a RADIUS server but requires configuring a static route for the RADIUS server. The Cisco ACS sample configuration procedure below uses an APIC in-band IP address.

Step 2 Create the **RADIUS Provider Group**.

a) In the **Navigation** pane, right-click the the**RADIUS Provider Groups** option, and select **Create RADIUS Provider Group**

b) Specify the RADIUS Provider Group name, description, and provider(s) as appropriate.

Step 3 Create the **Login Domain** for RADIUS.

a) In the **Navigation** pane, click the + icon to expand the**AAA Authentication** option.

b) In the **Navigation** pane, right-click the **Login Domains** option, and select **Create Login Domain**.

c) Specify the Login Domain name, description, realm, and provider group as appropriate.

What to Do Next

This completes the APIC RADIUS configuration steps. Next, configure the RADIUS server.

Configuring A Cisco Secure Access Control Server for RADIUS and TACACS+ Access to the APIC

Before You Begin

- The Cisco Secure Access Control Server (ACS) version 5.5 is installed and online.

Note ACS v5.5 was used to document these steps. Other versions of ACS might support this task but the GUI procedures might vary accordingly.

- The APIC RADIUS or TACACS+ keys are available (or keys for both if both will be configured).
- The APIC controllers are installed and online; the APIC cluster is formed and healthy.
- The RADIUS or TACACS+ port, authorization protocol, and key are available.

Procedure

Step 1 Log in to the ACS server to configure the APIC as a client.

a) Navigate to **Network Resources** > **Network Devices Groups** > **Network Devices and AAA Clients**.

b) Specify the client name, the APIC in-band IP address, select the TACACS+ or RADIUS (or both) authentication options.

Note If the only RADIUS or TACACS+ authentication is needed, select only the needed option.

c) Specify the authentication details such as Shared Secret (key), and port as appropriate for the authentication option(s).

Note The **Shared Secret**(s) must match the APIC **Provider** key(s).

Step 2 Create the Identity Group.

a) Navigate to **Users and Identity Stores** > **Internal Groups** option.

b) Specify the **Name**, and **Parent Group** as appropriate.

Step 3 Map users to the Identity Group.

a) In the **Navigation** pane, click the **Users and Identity Stores** > **Internal Identity Stores** > **Users** option.

b) Specify the user **Name**, and **Identity Group** as appropriate.

Step 4 Create the Policy Element.

a) Navigate to the **Policy Elements** option.

b) For RADIUS, specify the Authorization and Permissions > Network Access > Authorization Profiles **Name**. For TACACS+, specify the Authorization and Permissions > Device Administration > Shell Profile **Name**as appropriate.

c) For RADIUS, specify the **Attribute** as `cisco-av-pair`, **Type** as string, and the **Value** as shell:domain = <domain>/<role>/,<domain>// role as appropriate. For TACACS+, specify the **Attribute** as `cisco-av-pair`, **Requirement** as Mandatory, and the **Value** as shell:domain = <domain>/<role>/,<domain>// role as appropriate.

For example, if the *cisco-av-pair* is shell:domain = solar/admin/,common// read-all(16001), solar is the ACI tenant, admin is the role for this user that gives write privileges to this user in all of the tenant called solar, common is the ACI tenant common, read-all(16001) is the role with read privileges that gives this user read privileges to all of the ACI tenant common.

Step 5 Create a Service Selection Rule.

a) For RADIUS, create a service selection rule to associate the Identity Group with the Policy Element by navigating to **Access Policies** > **Default Device Network Access Identity** > **Authorization** and specifying the rule **Name**, **Status**, and **Conditions** as appropriate, and **Add** the `Internal Users:UserIdentityGroup in ALL Groups:<identity group name>`.

b) For TACACS+, create a service selection rule to associate the Identity Group with the Shell Profile by navigating to **Access Policies** > **Default Device Admin Identity** > **Authorization**. Specify the rule **Name**, **Conditions**, and **Select** the **Shell Profile** as appropriate.

What to Do Next

Use the newly created RADIUS and TACACS+ users to login to the APIC. Verify that the users have access to the correct APIC security domain according to the assigned RBAC roles and privileges. The users should not have access to items that have not been explicitly permitted. Read and write access rights should match those configured for that user.

Configuring Windows Server 2008 LDAP for APIC Access

Before You Begin

- First, configure the LDAP server, then configure the APIC for LDAP access.
- The Microsoft Windows Server 2008 is installed and online.
- The Microsoft Windows Server 2008 Server Manager ADSI Edit tool is installed. To install ADSI Edit, follow the instructions in the Windows Server 2008 Server Manager help.
- `AciCiscoAVPair` attribute specifications: Common Name = AciCiscoAVPair, LDAP Display Name = AciCiscoAVPair, Unique X500 Object ID = 1.3.6.1.4.1.9.22.1, Description = AciCiscoAVPair, Syntax = Case Sensitive String.

Note For LDAP configurations, best practice is to use AciCiscoAVPair as the attribute string. This avoids problems related to the limitation of common LDAP servers that do not allow overlapping object identifiers (OID); that is, the ciscoAVPair OID is already in use.

- A Microsoft Windows Server 2008 user account is available that will enable the following:
  - Running ADSI Edit to add the `AciCiscoAVPair` attribute to the Active Directory (AD) Schema.
  - Configuring an Active Directory LDAP user to have `AciCiscoAVPair` attribute permissions.

Procedure

Step 1 Log in to an Active Directory (AD) server as a domain administrator.

Step 2 Add the `AciCiscoAVPair` attribute to the AD schema.

a) Navigate to **Start** > **Run**, type mmc and press **Enter**.
The Microsoft Management Console (MMC) opens.
b) Navigate to **File** > **Add/Remove Sanp-in** > **Add**.
c) In the **Add Standalonee Snap-in** dialog box, select the **Active Directory Schema** and click **Add**.
The MMC **Console** opens.
d) Right-click the **Attributes** folder, select the **Create Attribute** option.
The **Create New Attribute** dialog box opens.
e) Enter AciCiscoAVPair for the **Common Name** , AciCiscoAVPair for the **LDAP Display Name**, 1.3.6.1.4.1.9.22.1 for the **Unique X500 Object ID**, and select Case Sensitive String for the **Syntax**.
f) Click **OK** to save the attribute.

Step 3 Update the **User Properties** class to include the **CiscoAVPair** attribute.

a) In the MMC **Console**, expand the **Classes** folder, right-click the **user** class, and choose **Properties**.
The **user Properties** dialog box opens.
b) Click the **Attributes** tab, select CiscoAVPair from the **Optional** list, and click **Add**.
The **Select Schema Object** dialog box opens.
c) In the **Select a schema object:** list, choose CiscoAVPair, and click **Apply**.
d) In the MMC **Console**, right-click the **Active Directory Schema**, and select **Reload the Schema**.

Step 4 Configure the AciCiscoAVPair attribute permissions.
Now that the LDAP includes the AciCiscoAVPair attributes, LDAP users need to be granted APIC permission by assigning them APIC RBAC roles.

a) In the ADSI Edit dialog box, locate a user who needs access to the APIC.
b) Right-click on the user name, and choose **Properties**.
The **<user> Properties** dialog box opens.
c) Click the **Attribute Editor** tab, select the *AciCiscoAVPair* attribute, and enter the *Value* as shell:domains = <domain>/<role>/,<domain>// role.
For example, if the *AciCiscoAVPair* is shell:domains = solar/admin/,common// read-all(16001), solar is the ACI tenant, admin is the role for this user that gives write privileges to this user in all of the tenant called solar, common is the ACI tenant common, read-all(16001) is the role with read privileges that gives this user read privileges to all of the ACI tennant common.
d) Click **OK** to save the changes and close the **<user> Properties** dialog box.

The LDAP server is configured to access the APIC.

What to Do Next

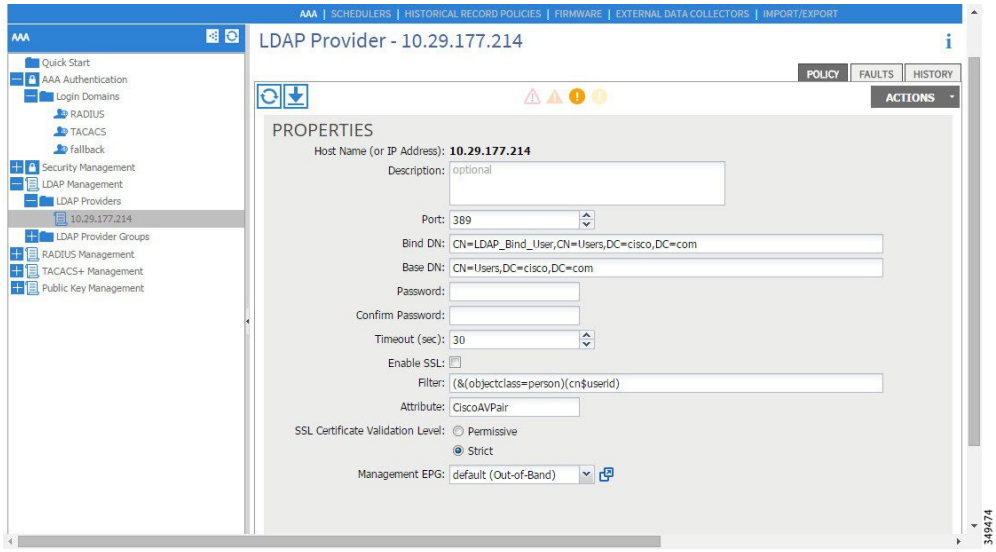
Configure the APIC for LDAP access.

Configuring APIC for LDAP Access

Before You Begin

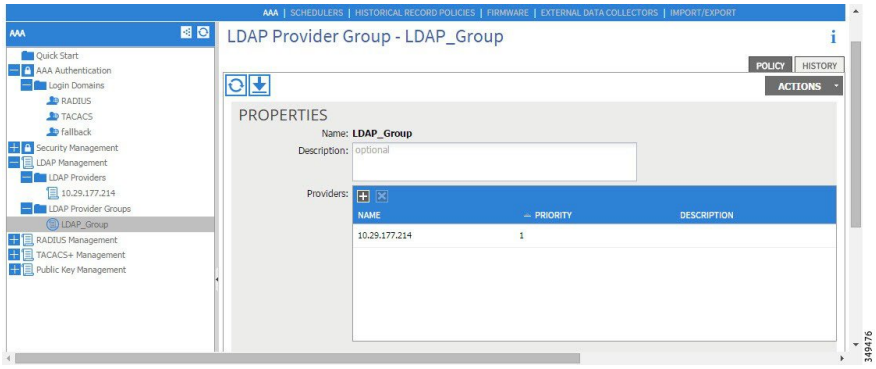
- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- The LDAP server host name or IP address, port, Bind DN, Base DN, and password are available.
- The APIC management EPG is available.
- An APIC user account is available that will enable the following:
  - Creating the LDAP provider and LDAP provider group.

Figure 7: LDAP Provider



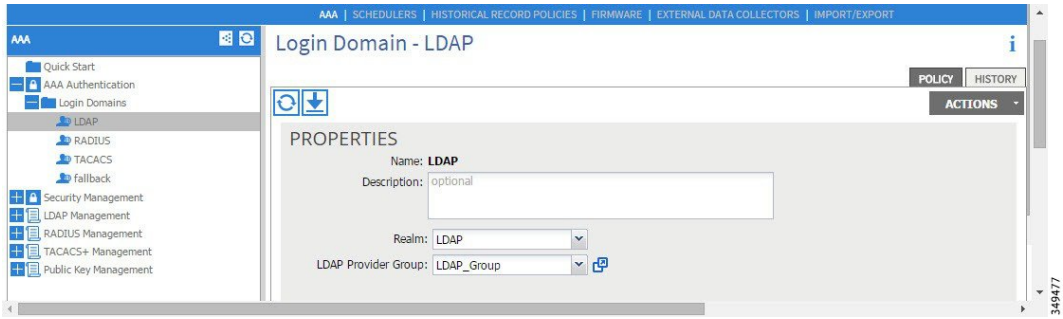
Note The Bind DN is the string that the APIC uses to log in to the LDAP server. The APIC uses this account to validate the remote user attempting to log in. The Base DN is the container name and path in the LDAP server where the APIC searches for the remote user account. This is where the password is validated. Filter is used to locate the attribute that the APIC requests to use for the *cisco-av-pair*. This contains the user authorization and assigned RBAC roles for use on the APIC. The APIC requests the attribute from the LDAP server.

Figure 8: LDAP Provider Group



◦ Creating the LDAP login domain.

Figure 9: AAA Login Domain for LDAP



Procedure

Step 1 On the APIC, configure the LDAP Provider.

a) On the APIC menu bar, click **ADMIN** > **AAA**.
b) In the **Navigation** pane, click the + icon to expand the**LDAP Managment** option.
c) In the **Navigation** pane, right-click the the**LDAP Providers** option, and select **Create LDAP Provider**.
d) Specify the LDAP host name (or IP address), port, Bind DN, Base DN, password, and management EPG.

Note If the APIC is configured for in-band management connectivity, choosing an out-of-band management EPG for LDAP access does not take effect. Alternatively, an out-of-band over an in-band management EPG can connect a LDAP server but requires configuring a static route for the LDAP server. The sample configuration procedures in this document use an APIC in-band management EPG.

Step 2 On the APIC, configure the LDAP Provider Group.

a) In the **Navigation** pane, right-click the the**LDAP Provider Groups** option, and select **Create LDAP Provider Group**

b) Specify the LDAP Provider Group name, description, and provider(s) as appropriate.

Step 3 On the APIC, configure the Login Domain for LDAP.

a) In the **Navigation** pane, click the + icon to expand the**AAA Authentication** option.

b) In the **Navigation** pane, right-click the **Login Domains** option, and select **Create Login Domain**.

c) Specify the Login Domain name, description, realm, and provider group as appropriate.

What to Do Next

This completes the APIC LDAP configuration steps. Next, test the APIC LDAP login access.

Changing the Default Behavior for Remote Users with Missing or Bad Cisco AV Pairs

Procedure

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

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

Step 3 In the **Work** pane, in the **Properties** area, from the **Remote user login policy** drop-down list, choose **Assign Default Role**.
The default value is **No Login**. The **Assign Default Role** option assigns the minimal read-only privileges to users that have missing or bad Cisco AV Pairs. Bad AV Pairs are those AV Pairs that fail the parsing rules.

Changing Default Behavior for Remote Users with Missing or Bad Cisco AV Pairs Using the NX-OS Style CLI

To change the default behavior for remote users with missing or bad Cisco AV pairs using the NX-OS CLI:

Procedure

Step 1 In the NX-OS CLI, start in Configuration mode.

Example:

```
apic1#
apic1# configure
```

Step 2 Configure the aaa user default role.

Example:

```
apic1(config)# aaa user default-role
 assign-default-role  assign-default-role
 no-login             no-login
```

Step 3 Configure the aaa authentication login methods.

Example:

```
apic1(config)# aaa authentication
 login  Configure methods for login

apic1(config)# aaa authentication login
 console  Configure console methods
 default  Configure default methods
 domain   Configure domain methods

apic1(config)# aaa authentication login console
 <CR>

apic1(config)# aaa authentication login domain
 WORD      Login domain name
 fallback
```

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.

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]
```

Configuring a Local User

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:
        {
            "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 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+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 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+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 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')
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.

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/log-out 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.

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` MO tracks the changes users make to objects and when the changes occurred.

Both `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.

Note 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 will give you 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-1 -> history -> audit log section, The GUI => History => Log options enable viewing event logs for a specific object identified in the GUI context.

The standard syslog, callhome, REST query, and CLI export mechanism 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 statics to a server.