Cisco ACI Guide

What is Cisco ACI ?

Application Centric Infrastructure (ACI)

The Cisco Application Centric Infrastructure (ACI) allows application requirements to define the network. This architecture simplifies, optimizes, and accelerates the entire application deployment life cycle.

Application Policy Infrastructure Controller (APIC)

The APIC manages the scalable ACI multi-tenant fabric. The APIC provides a unified point of automation and management, policy programming, application deployment, and health monitoring for the fabric. The APIC, which is implemented as a replicated synchronized clustered controller, optimizes performance, supports any application anywhere, and provides unified operation of the physical and virtual infrastructure.

The APIC enables network administrators to easily define the optimal network for applications. Data center operators can clearly see how applications consume network resources, easily isolate and troubleshoot application and infrastructure problems, and monitor and profile resource usage patterns.

The Cisco Application Policy Infrastructure Controller (APIC) API enables applications to directly connect with a secure, shared, high-performance resource pool that includes network, compute, and storage capabilities.

ACI Fabric

The Cisco Application Centric Infrastructure (ACI) Fabric includes Cisco Nexus 9000 Series switches with the APIC to run in the leaf/spine ACI fabric mode. These switches form a "fat-tree" network by connecting each leaf node to each spine node; all other devices connect to the leaf nodes. The APIC manages the ACI fabric.

The ACI fabric provides consistent low-latency forwarding across high-bandwidth links (40 Gbps, with a 100-Gbps future capability). Traffic with the source and destination on the same leaf switch is handled locally, and all other traffic travels from the ingress leaf to the egress leaf through a spine switch. Although this architecture appears as two hops from a physical perspective, it is actually a single Layer 3 hop because the fabric operates as a single Layer 3 switch.

The ACI fabric object-oriented operating system (OS) runs on each Cisco Nexus 9000 Series node. It enables programming of objects for each configurable element of the system. The ACI fabric OS renders policies from the APIC into a concrete model that runs in the physical infrastructure. The concrete model is analogous to compiled software; it is the form of the model that the switch operating system can execute.

All the switch nodes contain a complete copy of the concrete model. When an administrator creates a policy in the APIC that represents a configuration, the APIC updates the logical model. The APIC then performs the intermediate step of creating a fully elaborated policy that it pushes into all the switch nodes where the concrete model is updated.

The APIC is responsible for fabric activation, switch firmware management, network policy configuration, and instantiation. While the APIC acts as the centralized policy and network management engine for the fabric, it is completely removed from the data path, including the forwarding topology. Therefore, the fabric can still forward traffic even when communication with the APIC is lost.

More information

Various resources exist to start learning ACI, here is a list of interesting articles from the community.

- Adam Raffe: Learning ACI
- Luca Relandini: ACI for dummies
- Cisco DevNet Learning Labs about ACI

Using the ACI modules

The Ansible ACI modules provide a user-friendly interface to managing your ACI environment using Ansible playbooks.

For instance ensuring that a specific tenant exists, is done using the following Ansible task using the aci_tenant module:

```
- name: Ensure tenant customer-xyz exists
  aci_tenant:
    host: my-apic-1
    username: admin
    password: my-password

    tenant: customer-xyz
    description: Customer XYZ
    state: present
```

A complete list of existing ACI modules is available on the content tab of the ACI collection on Ansible Galaxy.

If you want to learn how to write your own ACI modules to contribute, look at the :ref:`Developing Cisco ACI modules <aci_dev_guide>` section.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst`, line 70); *backlink*
>
> Unknown interpreted text role "ref".

Querying ACI configuration

A module can also be used to query a specific object.

```
- name: Query tenant customer-xyz
  aci_tenant:
    host: my-apic-1
    username: admin
```

```
    password: my-password

    tenant: customer-xyz
    state: query
  register: my_tenant
```

Or query all objects.

```
- name: Query all tenants
  aci tenant:
    host: my-apic-1
    username: admin
    password: my-password

    state: query
  register: all_tenants
```

After registering the return values of the aci_tenant task as shown above, you can access all tenant information from variable `all_tenants`.

Running on the controller locally

As originally designed, Ansible modules are shipped to and run on the remote target(s), however the ACI modules (like most network-related modules) do not run on the network devices or controller (in this case the APIC), but they talk directly to the APIC's REST interface.

For this very reason, the modules need to run on the local Ansible controller (or are delegated to another system that *can* connect to the APIC).

Gathering facts

Because we run the modules on the Ansible controller gathering facts will not work. That is why when using these ACI modules it is mandatory to disable facts gathering. You can do this globally in your `ansible.cfg` or by adding `gather_facts: no` to every play.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 116)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: yaml
   :emphasize-lines: 3

    - name: Another play in my playbook
      hosts: my-apic-1
      gather_facts: no
      tasks:
      - name: Create a tenant
        aci_tenant:
          ...
```

Delegating to localhost

So let us assume we have our target configured in the inventory using the FQDN name as the `ansible_host` value, as shown below.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 131)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: yaml
   :emphasize-lines: 3

    apics:
      my-apic-1:
        ansible_host: apic01.fqdn.intra
        ansible_user: admin
        ansible_password: my-password
```

One way to set this up is to add to every task the directive: `delegate_to: localhost`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 142)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: yaml
   :emphasize-lines: 8

    - name: Query all tenants
      aci_tenant:
        host: '{{ ansible_host }}'
        username: '{{ ansible_user }}'
        password: '{{ ansible_password }}'

        state: query
      delegate_to: localhost
      register: all_tenants
```

If one would forget to add this directive, Ansible will attempt to connect to the APIC using SSH and attempt to copy the module and run it remotely. This will fail with a clear error, yet may be confusing to some.

Using the local connection method

Another option frequently used, is to tie the `local` connection method to this target so that every subsequent task for this target will use the local connection method (hence run it locally, rather than use SSH).

In this case the inventory may look like this:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 164)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: yaml
   :emphasize-lines: 6

    apics:
      my-apic-1:
        ansible_host: apic01.fqdn.intra
        ansible_user: admin
        ansible_password: my-password
        ansible_connection: local
```

But used tasks do not need anything special added.

```
- name: Query all tenants
  aci_tenant:
    host: '{{ ansible_host }}'
    username: '{{ ansible user }}'
    password: '{{ ansible_password }}'

    state: query
  register: all_tenants
```

Hint

For clarity we have added `delegate_to: localhost` to all the examples in the module documentation. This helps to ensure first-time users can easily copy&paste parts and make them work with a minimum of effort.

Common parameters

Every Ansible ACI module accepts the following parameters that influence the module's communication with the APIC REST API:

host
: Hostname or IP address of the APIC.

port
: Port to use for communication. (Defaults to `443` for HTTPS, and `80` for HTTP)

username
: User name used to log on to the APIC. (Defaults to `admin`)

password
: Password for `username` to log on to the APIC, using password-based authentication.

private_key
: Private key for `username` to log on to APIC, using signature-based authentication. This could either be the raw private key content (include header/footer) or a file that stores the key content. *New in version 2.5*

certificate_name
: Name of the certificate in the ACI Web GUI. This defaults to either the `username` value or the `private_key` file base name). *New in version 2.5*

timeout
: Timeout value for socket-level communication.

use_proxy
: Use system proxy settings. (Defaults to `yes`)

use_ssl
: Use HTTPS or HTTP for APIC REST communication. (Defaults to `yes`)

validate_certs
: Validate certificate when using HTTPS communication. (Defaults to `yes`)

output_level
: Influence the level of detail ACI modules return to the user. (One of `normal`, `info` or `debug`) *New in version 2.5*

Proxy support

By default, if an environment variable `<protocol>_proxy` is set on the target host, requests will be sent through that proxy. This behaviour can be overridden by setting a variable for this task (see :ref:`playbooks_environment`), or by using the `use_proxy` module parameter.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 234); *backlink*

Unknown interpreted text role "ref".

HTTP redirects can redirect from HTTP to HTTPS so ensure that the proxy environment for both protocols is correctly configured.

If proxy support is not needed, but the system may have it configured nevertheless, use the parameter `use_proxy: no` to avoid accidental system proxy usage.

Hint

Selective proxy support using the `no_proxy` environment variable is also supported.

Return values

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 246)

Unknown directive type "versionadded".

```
.. versionadded:: 2.5
```

The following values are always returned:

current
: The resulting state of the managed object, or results of your query.

The following values are returned when `output_level: info`:

previous
: The original state of the managed object (before any change was made).

proposed
: The proposed config payload, based on user-supplied values.

sent
: The sent config payload, based on user-supplied values and the existing configuration.

The following values are returned when `output_level: debug` or `ANSIBLE_DEBUG=1`:

filter_string
: The filter used for specific APIC queries.

method
: The HTTP method used for the sent payload. (Either `GET` for queries, `DELETE` or `POST` for changes)

response
: The HTTP response from the APIC.

status
: The HTTP status code for the request.

url
: The url used for the request.

Note

The module return values are documented in detail as part of each module's documentation.

More information

Various resources exist to start learn more about ACI programmability, we recommend the following links:

- :ref:`Developing Cisco ACI modules <aci_dev_guide>`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 288); *backlink*

  Unknown interpreted text role "ref".

- Jacob McGill: Automating Cisco ACI with Ansible
- Cisco DevNet Learning Labs about ACI and Ansible

ACI authentication

Password-based authentication

If you want to log on using a username and password, you can use the following parameters with your ACI modules:

```
username: admin
password: my-password
```

Password-based authentication is very simple to work with, but it is not the most efficient form of authentication from ACI's point-of-view as it requires a separate login-request and an open session to work. To avoid having your session time-out and requiring another login, you can use the more efficient Signature-based authentication.

Note

Password-based authentication also may trigger anti-DoS measures in ACI v3.1+ that causes session throttling and results in HTTP 503 errors and login failures.

Warning

Never store passwords in plain text.

The "Vault" feature of Ansible allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plain text in your playbooks or roles. These vault files can then be distributed or placed in source control. See :ref:`playbooks_vault` for more

information.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 313); *backlink*

Unknown interpreted text role "ref".

Signature-based authentication using certificates

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 319)

Unknown directive type "versionadded".

```
.. versionadded:: 2.5
```

Using signature-based authentication is more efficient and more reliable than password-based authentication.

Generate certificate and private key

Signature-based authentication requires a (self-signed) X.509 certificate with private key, and a configuration step for your AAA user in ACI. To generate a working X.509 certificate and private key, use the following procedure:

```
$ openssl req -new -newkey rsa:1024 -days 36500 -nodes -x509 -keyout admin.key -out admin.crt -subj '/CN=Admin/O=Your
```

Configure your local user

Perform the following steps:

- Add the X.509 certificate to your ACI AAA local user at :guilabel:`ADMIN` Â» :guilabel:`AAA`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 335); *backlink*

  Unknown interpreted text role "guilabel".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 335); *backlink*

  Unknown interpreted text role "guilabel".

- Click :guilabel:`AAA Authentication`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 336); *backlink*

  Unknown interpreted text role "guilabel".

- Check that in the :guilabel:`Authentication` field the :guilabel:`Realm` field displays :guilabel:`Local`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 337); *backlink*

  Unknown interpreted text role "guilabel".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 337); *backlink*

  Unknown interpreted text role "guilabel".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 337); *backlink*

  Unknown interpreted text role "guilabel".

- Expand :guilabel:`Security Management` Â» :guilabel:`Local Users`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 338); *backlink*

  Unknown interpreted text role "guilabel".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs]

[docsite][rst][scenario_guides]guide_aci.rst, line 338); *backlink*

Unknown interpreted text role "guilabel".

- Click the name of the user you want to add a certificate to, in the :guilabel:`User Certificates` area

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 339); *backlink*

Unknown interpreted text role "guilabel".

- Click the :guilabel:`+` sign and in the :guilabel:`Create X509 Certificate` enter a certificate name in the :guilabel:`Name` field

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 340); *backlink*

Unknown interpreted text role "guilabel".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 340); *backlink*

Unknown interpreted text role "guilabel".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 340); *backlink*

Unknown interpreted text role "guilabel".

  - If you use the basename of your private key here, you don't need to enter `certificate_name` in Ansible

- Copy and paste your X.509 certificate in the :guilabel:`Data` field.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 344); *backlink*

Unknown interpreted text role "guilabel".

You can automate this by using the following Ansible task:

```
- name: Ensure we have a certificate installed
  aci_aaa_user_certificate:
    host: my-apic-1
    username: admin
    password: my-password

    aaa_user: admin
    certificate_name: admin
    certificate: "{{ lookup('file', 'pki/admin.crt') }}"  # This will read the certificate data from a local file
```

Note

Signature-based authentication only works with local users.

Use signature-based authentication with Ansible

You need the following parameters with your ACI module(s) for it to work:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 367)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: yaml
   :emphasize-lines: 2,3

    username: admin
    private_key: pki/admin.key
    certificate_name: admin  # This could be left out !
```

or you can use the private key content:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 376)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: yaml
   :emphasize-lines: 2,3

    username: admin
    private_key: |
```

```
            -----BEGIN PRIVATE KEY-----
            <<your private key content>>
            -----END PRIVATE KEY-----
        certificate_name: admin  # This could be left out !
```

Hint

If you use a certificate name in ACI that matches the private key's basename, you can leave out the `certificate_name` parameter like the example above.

Using Ansible Vault to encrypt the private key

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 392)

Unknown directive type "versionadded".

```
.. versionadded:: 2.8
```

To start, encrypt the private key and give it a strong password.

```
ansible-vault encrypt admin.key
```

Use a text editor to open the private-key. You should have an encrypted cert now.

```
$ANSIBLE VAULT;1.1;AES256
56484318584354658465121889743213151843149454864654151618131547984132165489484654
45641818198456456489479874513215489484843614848456466655432455488484654848489498
....
```

Copy and paste the new encrypted cert into your playbook as a new variable.

```
private_key: !vault |
      $ANSIBLE VAULT;1.1;AES256
      56484318584354658465121889743213151843149454864654151618131547984132165489484654
      45641818198456456489479874513215489484843614848456466655432455488484654848489498
      ....
```

Use the new variable for the private_key:

```
username: admin
private_key: "{{ private_key }}"
certificate_name: admin  # This could be left out !
```

When running the playbook, use "--ask-vault-pass" to decrypt the private key.

```
ansible-playbook site.yaml --ask-vault-pass
```

More information

- Detailed information about Signature-based Authentication is available from Cisco APIC Signature-Based Transactions.
- More information on Ansible Vault can be found on the :ref:`Ansible Vault <vault>` page.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 437); *backlink*

Unknown interpreted text role "ref".

Using ACI REST with Ansible

While already a lot of ACI modules exists in the Ansible distribution, and the most common actions can be performed with these existing modules, there's always something that may not be possible with off-the-shelf modules.

The aci_rest module provides you with direct access to the APIC REST API and enables you to perform any task not already covered by the existing modules. This may seem like a complex undertaking, but you can generate the needed REST payload for any action performed in the ACI web interface effortlessly.

Built-in idempotency

Because the APIC REST API is intrinsically idempotent and can report whether a change was made, the aci_rest module automatically inherits both capabilities and is a first-class solution for automating your ACI infrastructure. As a result, users that require more powerful low-level access to their ACI infrastructure don't have to give up on idempotency and don't have to guess whether a change was performed when using the aci_rest module.

Using the aci_rest module

The aci_rest module accepts the native XML and JSON payloads, but additionally accepts inline YAML payload (structured like JSON). The XML payload requires you to use a path ending with `.xml` whereas JSON or YAML require the path to end with `.json`.

When you're making modifications, you can use the POST or DELETE methods, whereas doing just queries require the GET method.

For instance, if you would like to ensure a specific tenant exists on ACI, these below four examples are functionally identical:

XML (Native ACI REST)

```
- aci_rest:
    host: my-apic-1
    private_key: pki/admin.key

    method: post
    path: /api/mo/uni.xml
    content: |
      <fvTenant name="customer-xyz" descr="Customer XYZ"/>
```

JSON (Native ACI REST)

```
- aci_rest:
    host: my-apic-1
    private_key: pki/admin.key

    method: post
    path: /api/mo/uni.json
    content:
      {
        "fvTenant": {
          "attributes": {
            "name": "customer-xyz",
            "descr": "Customer XYZ"
          }
        }
      }
```

YAML (Ansible-style REST)

```
- aci_rest:
    host: my-apic-1
    private_key: pki/admin.key

    method: post
    path: /api/mo/uni.json
    content:
      fvTenant:
        attributes:
          name: customer-xyz
          descr: Customer XYZ
```

Ansible task (Dedicated module)

```
- aci tenant:
    host: my-apic-1
    private_key: pki/admin.key

    tenant: customer-xyz
    description: Customer XYZ
    state: present
```

> **Hint**
>
> The XML format is more practical when there is a need to template the REST payload (inline), but the YAML format is more convenient for maintaining your infrastructure-as-code and feels more naturally integrated with Ansible playbooks. The dedicated modules offer a more simple, abstracted, but also a more limited experience. Use what feels best for your use-case.

More information

Plenty of resources exist to learn about ACI's APIC REST interface, we recommend the links below:

- The ACI collection on Ansible Galaxy
- APIC REST API Configuration Guide -- Detailed guide on how the APIC REST API is designed and used, incl. many examples
- APIC Management Information Model reference -- Complete reference of the APIC object model
- Cisco DevNet Learning Labs about ACI and REST

Operational examples

Here is a small overview of useful operational tasks to reuse in your playbooks.

Feel free to contribute more useful snippets.

Waiting for all controllers to be ready

You can use the below task after you started to build your APICs and configured the cluster to wait until all the APICs have come online. It will wait until the number of controllers equals the number listed in the `apic` inventory group.

```
- name: Waiting for all controllers to be ready
  aci_rest:
    host: my-apic-1
    private key: pki/admin.key
    method: get
    path: /api/node/class/topSystem.json?query-target-filter=eq(topSystem.role,"controller")
  register: topsystem
  until: topsystem|success and topsystem.totalCount|int >= groups['apic']|count >= 3
  retries: 20
  delay: 30
```

Waiting for cluster to be fully-fit

The below example waits until the cluster is fully-fit. In this example you know the number of APICs in the cluster and you verify each APIC reports a 'fully-fit' status.

```
- name: Waiting for cluster to be fully-fit
```

```
aci_rest:
  host: my-apic-1
  private_key: pki/admin.key
  method: get
  path: /api/node/class/infraWiNode.json?query-target-filter=wcard(infraWiNode.dn,"topology/pod-1/node-1/av")
register: infrawinode
until: >
  infrawinode|success and
  infrawinode.totalCount|int >= groups['apic']|count >= 3 and
  infrawinode.imdata[0].infraWiNode.attributes.health == 'fully-fit' and
  infrawinode.imdata[1].infraWiNode.attributes.health == 'fully-fit' and
  infrawinode.imdata[2].infraWiNode.attributes.health == 'fully-fit'
retries: 30
delay: 30
```

APIC error messages

The following error messages may occur and this section can help you understand what exactly is going on and how to fix/avoid them.

APIC Error 122: unknown managed object class 'polUni'
: In case you receive this error while you are certain your aci_rest payload and object classes are seemingly correct, the issue might be that your payload is not in fact correct JSON (for example, the sent payload is using single quotes, rather than double quotes), and as a result the APIC is not correctly parsing your object classes from the payload. One way to avoid this is by using a YAML or an XML formatted payload, which are easier to construct correctly and modify later.

APIC Error 400: invalid data at line '1'. Attributes are missing, tag 'attributes' must be specified first, before any other tag
: Although the JSON specification allows unordered elements, the APIC REST API requires that the JSON `attributes` element precede the `children` array or other elements. So you need to ensure that your payload conforms to this requirement. Sorting your dictionary keys will do the trick just fine. If you don't have any attributes, it may be necessary to add: `attributes: {}` as the APIC does expect the entry to precede any `children`.

APIC Error 801: property descr of uni/tn-TENANT/ap-AP failed validation for value 'A "legacy" network'
: Some values in the APIC have strict format-rules to comply to, and the internal APIC validation check for the provided value failed. In the above case, the `description` parameter (internally known as `descr`) only accepts values conforming to Regex: `[a-zA-Z0-9\\!#$%()*,-./:;@ _{|}~?&+]+`, in general it must not include quotes or square brackets.

Known issues

The aci_rest module is a wrapper around the APIC REST API. As a result any issues related to the APIC will be reflected in the use of this module.

All below issues either have been reported to the vendor, and most can simply be avoided.

Too many consecutive API calls may result in connection throttling
: Starting with ACI v3.1 the APIC will actively throttle password-based authenticated connection rates over a specific threshold. This is as part of an anti-DDOS measure but can act up when using Ansible with ACI using password-based authentication. Currently, one solution is to increase this threshold within the nginx configuration, but using signature-based authentication is recommended.

  NOTE: It is advisable to use signature-based authentication with ACI as it not only prevents connection-throttling, but also improves general performance when using the ACI modules.

Specific requests may not reflect changes correctly (#35401)
: There is a known issue where specific requests to the APIC do not properly reflect changed in the resulting output, even when we request those changes explicitly from the APIC. In one instance using the path `api/node/mo/uni/infra.xml` fails, where `api/node/mo/uni/infra/.xml` does work correctly.

  NOTE: A workaround is to register the task return values (for example, `register: this`) and influence when the task should report a change by adding: `changed_when: this.imdata != []`.

Specific requests are known to not be idempotent (#35050)
: The behaviour of the APIC is inconsistent to the use of `status="created"` and `status="deleted"`. The result is that when you use `status="created"` in your payload the resulting tasks are not idempotent and creation will fail when the object was already created. However this is not the case with `status="deleted"` where such call to an non-existing object does not cause any failure whatsoever.

  NOTE: A workaround is to avoid using `status="created"` and instead use `status="modified"` when idempotency is essential to your workflow..

Setting user password is not idempotent (#35544)
: Due to an inconsistency in the APIC REST API, a task that sets the password of a locally-authenticated user is not idempotent. The APIC will complain with message `Password history check: user dag should not use previous 5 passwords`.

  NOTE: There is no workaround for this issue.

ACI Ansible community

If you have specific issues with the ACI modules, or a feature request, or you like to contribute to the ACI project by proposing changes or documentation updates, look at the Ansible Community wiki ACI page at:
https://github.com/ansible/community/wiki/Network:-ACI

You will find our roadmap, an overview of open ACI issues and pull-requests, and more information about who we are. If you have an interest in using ACI with Ansible, feel free to join! We occasionally meet online (on the #ansible-network chat channel, using Matrix at ansible.im or using IRC at irc.libera.chat) to track progress and prepare for new Ansible releases.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-`

devel\docs\docsite\rst\scenario_guides\[ansible-devel][docs][docsite][rst][scenario_guides]guide_aci.rst, line 646)

Unknown directive type "seealso".

```
.. seealso::

   `ACI collection on Ansible Galaxy <https://galaxy.ansible.com/cisco/aci>`_
       View the content tab for a complete list of supported ACI modules.
   :ref:`Developing Cisco ACI modules <aci_dev_guide>`
       A walkthrough on how to develop new Cisco ACI modules to contribute back.
   `ACI community <https://github.com/ansible/community/wiki/Network:-ACI>`_
       The Ansible ACI community wiki page, includes roadmap, ideas and development documentation.
   :ref:`network_guide`
       A detailed guide on how to use Ansible for automating network infrastructure.
   `Network Working Group <https://github.com/ansible/community/tree/master/group-network>`_
       The Ansible Network community page, includes contact information and meeting information.
   `User Mailing List <https://groups.google.com/group/ansible-project>`_
       Have a question?  Stop by the google group!
```