Using the APIC CLI

Accessing the Object Model CLI

Note From Cisco APIC Release 1.0 until Release 1.2, the Object Model CLI was the default CLl, appearing when you logged in to APIC using SSH. Beginning with Cisco APIC Release 1.2, the default CLI is the NX-OS style CLI.

Procedure

Step 1 From a secure shell (SSH) client, open an SSH connection to APIC at username@ip-address. Use the administrator login name and the out-of-band management IP address that you configured during the initial setup. For example, admin@192.168.10.1.

Step 2 When prompted, enter the administrator password.

Step 3 At the command line prompt, type **bash**.

Example

This example shows how to reach the object model CLI from the initial CLI prompt.

```
apic1# bash
admin@apic1:~>
```

Viewing Managed Objects

Use the **cat summary** command to display a summary of the managed object (MO) in a given context within the Management Information Tree (MIT):

Note You can also use the **less** and **more** commands to displayMO files one screen at a time.

```
admin@apic1:common> cat summary
name               : common
description        :
tags               : uni/tn-common
ownerkey           :
ownertag           :
alias              :
monitoring-policy  :
epg-address-pool   :

security-domains:
name    description
------  -----------
common
```

Navigating the Management Information Tree

The Management Information Tree (MIT) contains a variety of scopes, including:

- aaa
- auditlog
- controller
- eventlog
- fabric-policies
- faults
- faults-history
- firmware
- health
- health-history
- import-export
- l4-l7-inventory
- l4-l7-packages

- local-user
- pod
- schedulers
- security-domains
- switch
- tenant
- trafficmap
- version
- vm-inventory
- vm-policies

To navigate quickly through these scopes, you can use the following commands:

- scope—Jumps to the directory for a context.
- show—Displays the summary for a context.
- where—Displays the management information tree (MIT) directory path for a context.

For more information about these commands, see Command Reference

MO Browser Utility

The APIC CLI contains a managed object (MO) browser utility for viewing and editing MOs with a interface similar to vi. For more information about mobrowser, see mobrowser.

Entering a Configuration

You can use the **moconfig**, **moset**, and **modelete** commands to create a configuration.

The **moconfig** command creates a new context by name, whereas **moset** sets properties on an existing MO. The **modelete** command removes a scope by name, typically a sub-scope.

To override default settings, you can specify additional properties with the **mocreate** command. If you want to override default settings for a context, you can specify additional properties with the mocreate command. For more information, see mocreate.

Note You can also use the APIC GUI, REST API, or Python API to enter a configuration. For more information about these tools, see the *APIC Getting Started Guide* and the *APIC Python API and SDK*.

Displaying Command Differences

The **moconfig diff** command summarizes any unsaved changes are present in the configuration buffer. You can use the **moconfig commit** command to apply the new properties to the MO.

Using Configuration Wizards

Wizards simplify the process of creating a configuration. When you run a wizard in a given context (such as tenants), the wizard helps you create a complete configuration within a given context (for example, tenants or private networks).

Launching a Wizard

To start a wizard, run the .wiz file. For example, the tenant context provides a wizard that you can run using the **./tenant.wiz** Bash command.

Wizard Options

Wizards support command completion. You can enter **?** to list the available options.

```
description       : MyCompany BD
network           : ?
 default          network
 inb              network
 overlay-1        network
network           : inb
```

Example

The following example shows the full output of the tenant wizard.

```
admin@apic1:tenants> ./tenant.wiz

tenant
------
name              : MyCompany
alias             : MyCompany_tenant
description       : This is MyCompany
monitoring-policy : default

private-network
---------------
name              : MyCompany_net
description       : MyCompany Network
bgp-timers        : default
ospf-timers       : default
monitoring-policy : default

bridge-domain
---------------
name              : MyCompany_domain
description       : MyCompany BD
network           : ?
 default          network
 inb              network
 overlay-1        network
network           : inb

Do you want to create another private-network (y/n): n
```

```
Do you want to view the corresponding commands? (y/n): y
--------------------------------------------------------------------------------

mocreate MyCompany
cd MyCompany
moset alias MyCompany_tenant
moset description This is MyCompany
moset monitoring-policy default
cd /aci/tenants/MyCompany/networking
cd /aci/tenants/MyCompany/networking/private-networks
mocreate MyCompany_net
cd MyCompany_net
moset description MyCompany Network
moset bgp-timers default
moset ospf-timers default
moset monitoring-policy default
cd /aci/tenants/MyCompany/networking/bridge-domains
mocreate MyCompany_domain
cd MyCompany_domain
moset description MyCompany BD
moset network inb
cd /aci/tenants/MyCompany/networking/private-networks/MyCompany_net
cd /aci/tenants/MyCompany/networking
cd /aci/tenants/MyCompany

Do you want to commit changes? (y/n): y

Committing all the mos...
Committed mo tenants/MyCompany
Committed mo tenants/MyCompany/networking/private-networks/MyCompany_net
Committed mo tenants/MyCompany/networking/bridge-domains/MyCompany_domain
done
admin@apic1:tenants>
```

Skipping Properties

You can use the **Ctrl+N** command to skip options within a wizard.

Note Wizards dynamically track missing properties. If you skip a property, you can run the appropriate wizard to complete the configuration later. For example, if you run the tenant wizard, you can skip properties within the private-network context:

```
admin@apic1:tenants> ./tenant.wiz
<output truncated>
private-network
---------------
name               : Company_net
description        : s...skipping
bgp-timers         : s...skipping
ospf-timers        : s...skipping
monitoring-policy  : s...skipping

bridge-domain
---------------

name               : default

<output truncated>
```

Later, you can run the private-network wizard later to complete the configuration.

```
admin@apic1:networking> ls
bridge-domains             external-routed-networks  fv-tenant-common  fv-tenant-mgmt
private-network.wiz  protocol-policies
external-bridged-networks  fv-tenant-MyCompany            fv-tenant-infra   fv-tenant-test
```

```
  private-networks
admin@apic1:networking> ./private-network.wiz
```

Creating Configuration Templates

Configuration templates allow you to create reusable network configurations that you can apply using orchestration tools, shell scripts, and other tools. The following sections describe how to use the APIC CLI to create configuration templates.

Creating Templates Using the moconfig Command

The **moconfig** command simplifies the process of creating configuration templates. When you create a configuration using the GUI, CLI, or API, you can use the **moconfig running** command to display the resulting configuration in a given context.

For example, you can use the GUI to create a tenant configuration including the following properties:

- Name
- Alias
- Description
- Tags
- Monitoring Policy
- Security Domains

After you enter the configuration in the GUI, you can use the **moconfig** command in the new APIC context to display the commands that make up the configuration. For example, if you create a new tenant MyCompany, you can display the configuration commands as follows:

```
admin@apic1:tenants> ls
common  infra  mgmt  MyCompany  tenant.wiz
admin@apic1:tenants> cd MyCompany/
admin@apic1:MyCompany> moconfig running
cd /aci/viewfs/tenants
mocreate MyCompany
cd MyCompany
moset description 'My Company Network'
moset alias Home
moset monitoring-policy default
moconfig commit
cd networking
cd private-networks
mocreate local_net
cd local_net
moset description 'Local network'
moset bgp-timers default
moset ospf-timers default
moset monitoring-policy default
moconfig commit
cd ..
cd ..
cd bridge-domains
mocreate BD1
cd BD1
moset description 'Bridge domain 1'
moset custom-mac-address 00:22:BD:F8:19:FF
moset arp-flooding no
moset unicast-routing yes
```

```
moset network overlay-1
moconfig commit
cd ..
cd ..
cd ..
cd ..
admin@apic1:MyCompany>
```

For more information about using the **moconfig running** command, see the moconfig command. .

Creating Templates using Configuration Wizards

When running a configuration wizard, you can use the corresponding commands option to summarize the configuration created by the wizard. You can modify and replicate this configuration on other nodes or devices.

The following example shows how to display the command output from a configuration wizard.

Note The command output is truncated.

```
admin@apic1:tenants> ./tenant.wiz

<Output truncated>

Do you want to create another private-network (y/n): n

Do you want to view the corresponding commands? (y/n): y
--------------------------------------------------------------------------------
mocreate MyCompany
cd MyCompany
moset alias Home
moset description My Company Network
moset monitoring-policy default
cd /aci/tenants/MyCompany/networking
cd /aci/tenants/MyCompany/networking/private-networks
mocreate local_net
cd local_net
moset description Local network
moset bgp-timers default
moset ospf-timers default
moset monitoring-policy default
cd /aci/tenants/MyCompany/networking/bridge-domains
mocreate BD1
cd BD1
moset description Bridge domain 1
moset network overlay-1
cd /aci/tenants/MyCompany/networking/private-networks/local_net
cd /aci/tenants/MyCompany/networking
cd /aci/tenants/MyCompany
--------------------------------------------------------------------------------
<Output truncated>
```

For more information about using wizards, see Using Configuration Wizards.

Customizing Commands

The APIC CLI allows you to extend Linux commands in the Bash interface using YAML (.yml) files in the /etc/scopedefs directory. YAML configuration files specify Linux commands to run and available options at each scope.

You can use YAML files to create new commands and extend existing Linux commands. YAML files allow you to define custom interfaces for users by placing a unique .yml file in the user's scope in the MIT.

You can customize the following commands using YAML.

- show—Displays the APIC configuration in a format similar to Cisco IOS and NX-OS. For more information, see show.
- create— Executes a wizard within a given scope; the wizard creates relevant objects in the MIT. For more information, see create.
- where—Displays the directory for a context, such as tenant or l4-l7-services. For more information, see where.
- scope—To jump to the directory for a context, such as tenant or l4-l7-services. For more information, see scope.
- attach—Opens an SSH session to a specified fabric node. For more information, see attach.

Sample YAML Command Definitions

controller Command

The following example shows the **controller** command output:

```
admin@apic1:aci> controller

operational-cluster-size                                  : 3
differences-between-local-time-and-unified-cluster-time   : 0
administrative-cluster-size                               : 3

controllers:
id  name   ip        cluster-admin-state  cluster-operational-  health-state  up-time
     system-current-time
                                          state

--  -----  --------  -------------------  --------------------  ------------  ---------------
  ---------------------
1   apic1  10.0.0.1  in-service           available             fully-fit     62:02:38:00.000
  2014-05-

     01T21:40:46.120+00:00
2   apic2  10.0.0.2  in-service           available             fully-fit     62:02:38:00.000
  2014-05-

     01T21:40:46.211+00:00
3   apic3  10.0.0.3  in-service           available             fully-fit     62:02:38:00.000
  2014-05-

     01T21:40:46.263+00:00
```

The following example shows the YAML definition of the **controller** command:

```
- controller:
    help: 'Controller Node'
    type: alias
    dirFormat: '/aci/system/controllers/'
    fileType: 'summary'
    sub:
       - name: id
         label: id
         type: arg
         modelclass: fabric.Node
         modelprop: id
         classfilter: 'fabric.Node.role == "1"'
         dirFormat: '/aci/system/controllers/%(id)s'
         fileType: 'summary'
         help: 'controller'
```

tenant Command

The following example shows the **tenant** command output:

```
admin@apic1:~> show tenant infra bridge-domains default
# Executing command: cat /aci/tenants/infra/networking/bridge-domains/default/mo

# bridge-domain

# Naming properties (DO NOT EDIT):
# name                          : default

# Configurable Properties:
description                     :
custom-mac-address              : 00:22:BD:F8:19:FF
l2-unknown-unicast              : hardware-proxy
arp-flooding                    : no
unicast-routing                 : yes
ownerkey                        :
ownertag                        :
network                         : overlay-1
igmp-snoop-policy               :
end-point-retention-policy      :
l3-out                          :
external-route                  :
route-profile                   :
monitoring-policy               :
```

The following example shows an excerpt of the YAML definition of the **tenant** command:

```
- tenant:
    help: 'Tenant'
    type: alias
    dirFormat: '/aci/tenants/'
    fileType: 'summary'
    name: tenant
    sub:
      - name: name
        label: name
        type: arg
        modelclass: fv.Tenant
        modelprop: name
        dirFormat: '/aci/tenants/%(name)s'
        fileType: 'summary'
        help: Tenant name
        sub:
          - name: bridge-domains
            label: bridge-domains
            type: keyword
            dirFormat: '/aci/tenants/%(name)s/networking/bridge-domains/'
            fileType: 'summary'
            help: "All Bridge-domains"
            sub:
              - name: bd
                label: bridge-domain-name
                type: arg
                modelclass: fv.BD
                modelprop: name
                dirFormat: '/aci/tenants/%(name)s/networking/bridge-domains/%(b\d)s'
                fileType: 'mo'
                help: Bridge domain name
          - name: application-profiles
            label: application-profiles
            type: keyword
            dirFormat: '/aci/tenants/%(name)s/application-profiles/'
            fileType: 'summary'
            help: "All application profiles"
            sub:
              - name: ap
                label: application-profile-name
                type: arg
                modelclass: fv.Ap
                modelprop: name
                dirFormat: '/aci/tenants/%(name)s/application-profiles/%(ap)s'
```

```
      fileType: 'mo'
      help: Application profile name
  - name: private-networks
    label: private-networks
    type: keyword
    dirFormat: '/aci/tenants/%(name)s/networking/private-networks/'
    fileType: 'summary'
    help: "All private networks"
    sub:
      - name: pn
        label: private-network-name
        type: arg
        modelclass: fv.Ctx
        modelprop: name
        dirFormat: '/aci/tenants/%(name)s/networking/private-networks/%\(pn)s'
        fileType: 'mo'
        help: Private network name
        type: arg
        modelclass: fv.Ctx
        modelprop: name
        dirFormat: '/aci/tenants/%(name)s/networking/private-networks/%\(pn)s'
        fileType: 'mo'
        help: Private network name
        (...)
```

YAML File Format

File Format

You can use the following keywords to define using custom command a .yml file.

- **help—**A help string that defines the function of the command, argument, or keyword, as follows: **help: 'Displays faults for the current path.'**
- **type—**Specifies one of the following command actions:
  - **alias—**Similar to a standard Unix alias command. References a directory in the MIT.
  - **command—**Executes a unix command, such as **cat** or **version**.
  - **showcmd—**Executes a show option within a configuration command, such as **firmware list**.
- **dirFormat—**Specifies the directory format for the scope. For example, aci/fabric/inventory/pod-1/node-%(id)s specifies a subdirectory for each node.

Note %(<arg>)s specifies an argument in the dirFormat and cmdFormat strings.

- **fileType—**Specifies a file type: you can specify **summary** or **mo**.
- **cmdFormat—**Defines the command to execute, as shown in the following example: **cmdFormat: 'eventlog'** You can specify that a command execute in a specific scope.
- The following options describe command arguments and keywords.
  - **sub—**Defines a sub-scope. Applies only to **alias** commands.
  - **name—**The name of the argument or keyword.
  - **label—**Defines a label for the argument or keyword.

- **type—**The sub-command parameter type. **arg** specifies an argument; **keyword** specifies a keyword.
-

- You can use the following options for autocompletion:
  - **classfilter—**Defines a class filter. For example, **classfilter: 'fabric.Node.role == "1"'** restricts results to MOs that have a role value of 1.
  - **fill—**Enter **fill: auto** to display child directories for a scope. Applies only to **alias** commands.
  - **modelclass—**Defines a scope used to autocomplete results.
  - **modelprop—**Defines a property used to autocomplete results, such as **name** or **id**.