Understanding the Command-Line Interface

About the Application Policy Infrastructure Controller

This guide describes how to use the command-line interface (CLI) of the Application Policy Infrastructure Controller (APIC), which consists of the standard Bash command language interpreter shell plus a set of custom commands for the APIC.

For detailed reference information about API classes, methods, and types, see the *Cisco APIC Management Information Model Reference*, which is a web-based application. To learn about the features and operation of the Application Policy Infrastructure Controller, see the available white papers and the *Cisco Application Centric Infrastructure Fundamentals*.

Configuration Options

The Cisco Application Policy Infrastructure Controller (APIC) offers the following configuration options:

- Direct Configuration with the Object Model CLI—You can use the Object Model CLI extensions to the BASH shell to directly manipulate managed objects (MO) and the Management Information Tree (MIT). This document provides information about direct configuration using the Object Model CLI.
- NX-OS Style CLI—Beginning with Cisco APIC Release 1.2, you can use NX-OS style CLI commands for configuration.

Note This document does not provide information about the APIC NX-OS style CLI interface. For information, see *Cisco APIC NX-OS Style Command-Line Interface Configuration Guide*.

- Shell Scripts— You can use the Bash shell to automate some tasks using shell scripting. For more information about Bash, see Understanding the GNU Bash Shell.
- Python API— Enables more extensive automation. For more information about the Python API, see the *Cisco APIC Python SDK Reference*.

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. The object model CLI is available by typing the **bash** command at the initial CLI prompt.

Understanding Managed Objects

The APIC system configuration and state are modeled as a collection of managed objects (MOs), which are abstract representations of a physical or logical entity that contain a set of configurations and properties. For example, servers, chassis, I/O cards, and processors are physical entities represented as MOs; resource pools, user roles, service profiles, and policies are logical entities represented as MOs.

At runtime all MOs are organized in a tree structure called the Management Information Tree, providing structured and consistent access to all MOs in the system.

Understanding the File System

The Management Information Tree (MIT) consists of hierarchically organized MOs that allow you to manage the APIC. Each MO is modeled as a Linux directory that contains all child MOs as subdirectories and all properties in an mo file.

Here is a sample output of the file system: the local-users directory contains subdirectories for three users: admin, john, and viewer.

```
admin@apic1:local-users> pwd
/home/admin/aci/admin/aaa/security-management/local-users
admin@apic1:local-users> ls -al
total 3
drw-rw---- 1 admin admin 512 Apr 10 16:58 .
drw-rw---- 1 root  root  512 Apr  8 07:06 ..
drw-rw---- 1 root  root  512 Apr  8 07:06 admin
drw-rw---- 1 admin admin 512 Jan 28 20:16 john
-r--r----- 1 admin admin 197 Apr 10 16:58 summary
```

Role based access controls (RBAC) allow you to grant permissions to a user so that the user can manage another user. In this case, admin and viewer users are owned by root, while john is owned by admin.

Note The absence of an mo file in this directory indicates that there are no configurable properties at this directory level.

```
admin@apic1:local-users> cd admin
admin@apic1:admin> pwd
/home/admin/aci/admin/aaa/security-management/local-users/admin
admin@apic1:admin> ls -al
total 4
```

```
drw-rw---- 1 admin admin 512 Jul 22 14:29 .
drw-rw---- 1 admin admin 512 Jul 22 14:29 ..
-rw-rw---- 1 admin admin 485 Jul 22 14:29 mo
drw-rw---- 1 admin admin 512 Jul 22 14:29 operational
drw-rw---- 1 admin admin 512 Jul 22 14:29 security-domains
drw-rw---- 1 admin admin 512 Jul 22 14:29 ssh-keys
-r--r----- 1 admin admin 493 Jul 22 14:29 summary
drw-rw---- 1 admin admin 512 Jul 22 14:29 user-certificates
```

Understanding the GNU Bash Shell

Bash (Bourne Again SHell) is a Unix shell or command-line interpreter supported by a variety of operating systems. You can use the Bash interface to directly configure the APIC or develop Bash shell scripts to automate tasks. Bash provides a variety of command line and scripting features.

Synopsis

Bash is an sh-compatible command language interpreter that executes commands read from the standard input or from a file. Bash also incorporates useful features from the Korn and C shells (ksh and csh). Bash is ultimately intended to be a faithful implementation of the IEEE POSIX Shell and Tools specification (IEEE Working Group 1003.2).

Bash supports a variety of features including:

- Command-line editing
- Unlimited size command history
- Job control
- Shell functions and aliases
- Indexed arrays of unlimited size
- Integer arithmetic in any base from 2 to 64

For more information about the Bash shell , see http://www.gnu.org/software/bash/bash.html.

Bash Extensions

The APIC includes following extensions of the Bash shell:

Networking Naming Conventions

Network operating systems typically use a forward slash (/) as a separator for interfaces, network addresses, and other settings. However, the Bash shell restricts the use of the forward slash in file names. While Bash provides for an escape character, the APIC file system simplifies network naming by using a colon (:) as a separator. The following examples describe how to use this separator.

Interface Naming

The APIC Bash extension uses the colon (:) character to delimit interface names. For example, the interface Ethernet 1/46 is written as **eth1:46**.

The following example shows output of interfaces on a node:

```
admin@apic1:physical-interfaces> pwd
/aci/fabric/inventory/fabric-pod-1/fabric-node-17/interfaces/physical-interfaces
admin@apic1:physical-interfaces> ls
eth1:1   eth1:17  eth1:24  eth1:31  eth1:39  eth1:46  eth1:53  eth1:60
eth1:10  eth1:18  eth1:25  eth1:32  eth1:4   eth1:47  eth1:54  eth1:7
eth1:11  eth1:19  eth1:26  eth1:33  eth1:40  eth1:48  eth1:55  eth1:8
eth1:12  eth1:2   eth1:27  eth1:34  eth1:41  eth1:49  eth1:56  eth1:9
eth1:13  eth1:20  eth1:28  eth1:35  eth1:42  eth1:5   eth1:57  summary
eth1:14  eth1:21  eth1:29  eth1:36  eth1:43  eth1:50  eth1:58
eth1:15  eth1:22  eth1:3   eth1:37  eth1:44  eth1:51  eth1:59
eth1:16  eth1:23  eth1:30  eth1:38  eth1:45  eth1:52  eth1:6
admin@apic1:physical-interfaces>
```

Network Address Naming

The APIC Bash extension uses the colon (:) character to delimit network addresses. For example, the network 192.168.1.0 and subnet 255.255.255.0 are written as follows:

```
192.168.1.0:255.255.255.0
```

Command Completion

The APIC provides tab completion for standard Linux commands and APIC-specific commands listed in the Command Reference. When you press the **Tab** key at the end of a command or option abbreviation, the CLI displays the command in full or the next available keyword or argument choice.

For example, you can use the tab key to display available directories:

```
admin@apic1:aci> cd tenants/ <Tab>
common/ infra/  mgmt/
```

Command History

The APIC CLI supports the Bash shell history functions. To display the command history, you can use the **Up Arrow** or **Down Arrow**, as well as the **history** command.

You can reenter a command in the history by stepping through the history to recall the desired command and pressing **Enter**. You can also recall a command and change it before you enter it.

In addition, you can directly search for a previous command by pressing **Ctrl-r** and then typing part of the desired command until the command is displayed.

For more information about the Bash shell including additional command history functions, see http://www.gnu.org/software/bash/bash.html

Command Help

The CLI provides two forms of context sensitive help:

- Inline help—At any time, you can enter the **Esc** key twice to display the options available at the current state of the command syntax. If you have not entered anything at the prompt, entering **Esc** key twice lists all available commands for the current command mode. If you have partially entered a command, entering **Esc** key twice lists all available keywords and arguments available at your current position in the command syntax.

- Man pages—At the command prompt, you can enter the **man** followed by a command or path to a managed object (MO) under /aci to display a UNIX-style man page. Man pages are not available for all commands or scopes.

Mount Points

The APIC CLI has three mount points: aci, mit, and debug. The following sections describe the mount points in more detail.

When you log into the APIC, the aci, debug, and mit mount points are displayed default directory:

```
admin@apic1:~> ls
aci  debug  mit
```

Note A link to each file system is provided in each user home directory.

The following sections describe the mount points in more detail.

aci Mount Point

The aci file system organizes MOs and properties into a concise format for interactive user sessions. The aci mount point is intended for most users and is the primary CLI interface for the APIC.

mit Mount Point

The Management Information Tree (MIT) file system allows advanced users to directly view and configure MOs within the MIT. The directory structure of the mitfs is the same as aci except that MOs are displayed as native MIT objects.

For example, the mit mount point displays the admin user as follows:

```
admin@apic1:user-admin> pwd
/mit/uni/userext/user-admin
admin@apic1:user-admin> ls -ltr
total 4
drw-rw---- 1 root  root  512 Jan 27 15:08 userdomain-all
drw-rw---- 1 root  root  512 Jan 27 15:08 userdata
-r--r----- 1 root  root  665 Jan 27 15:08 mo
drw-rw---- 1 admin admin 512 Jan 28 17:56 history
drw-rw---- 1 admin admin 512 Jan 28 17:56 faults
```

Note The mit mount point is intended for advanced users with a strong understanding of MO configuration.

debug Mount Point

The debug mount point allows you to view and debug configurations across multiple APIC, leaf, and spine devices. The debug mount point is intended for troubleshooting by advanced users.

Role-Based Access Control

With role-based access control (RBAC), you can limit access to device operations by assigning roles to users. You can customize access and restrict it to users who require it.

Applying Permissions and Security

Role-Based Access Control (RBAC) allows you to control user permissions by creating roles with a set of permissions and assigning them to users. RBAC allows you to apply permission to a user by assigning a role rather than directly configuring permissions.

Within the APIC CLI, you can grant permissions to users to manipulate specific parts of the Management Information Tree (MIT) such as a managed object (MO).

The following example shows how to use the **ls** command to display RBAC permissions within the APIC CLI. The command output displays files and UNIX read/write/execute file permissions and the time and date when the file was last modified.

```
admin@apic1:user-admin> ls -al
total 4
drw-rw---- 1 admin admin 512 Jul 22 14:25 .
drw-rw---- 1 admin admin 512 Jul 22 14:25 ..
-rw-rw---- 1 admin admin 421 Jul 22 14:25 mo
-r--r----- 1 admin admin 608 Jul 22 14:25 summary
drw-rw---- 1 admin admin 512 Jul 22 14:25 userdata
drw-rw---- 1 admin admin 512 Jul 22 14:25 userdomain-all
```

User Management

By default, each user is provided with a home directory at /home/<username>. This directory gives permissions for a user to create sub-directories and files. Files created within /home/<username> inherit the default umask permissions and are accessible by the user and the administrator (admin).

We recommend that users create a */userid* directory to store files- such as /home/jsmith -when logging in for the first time. Thereafter the APIC treats the */userid* directory as the user's home directory.