CHAPTER 3

Using the nrcmd Program as an API

You can use the **nrcmd** command to interactively configure and control a Cisco CNS Network Registrar cluster, or you can use it as a programming interface for another program or script. This chapter describes how to use the **nrcmd** command as an application programming interface (API).

Note With the Network Registrar 6.2 release, Cisco has made available a public API. The public API might better suit your business needs than **nrcmd**. Documentation for using that API is found on your installation media.

Connecting to Network Registrar

When you use the **nrcmd** command, you connect to a Network Registrar cluster to read and write configuration data, read state data, and perform control operations.

A Network Registrar cluster consists of:

- The data manager, the MCD server, which controls access to persistent datastores that contain configuration and state information for the DNS, DHCP, and TFTP servers.
- The server agent, Network Registrar Server Agent, which starts and stops the protocol servers, and provides a standard control interface to them.
- The DNS, DHCP, and TFTP protocol servers.

Performing Authentication

When you log in to a cluster you are authenticated with a name and a password. The authentication protocol uses one-way hashes so that a password does not travel across the wire. In interactive mode, the **nrcmd** command prompts for a valid username and password. You can also provide the username or password on the command line, in the environment.

Because you may not want to embed the administrator password in a command script, the environment variable entries provide alternate locations with different visibility levels. The environment variables CNR_CLUSTER, CNR_NAME, and CNR_PASSWORD specify the cluster, administrator name and administrator password values, respectively.

For details about creating administrator names and passwords, see the "admin" section on page 2-9.

Choosing Scripting Techniques

Because **nrcmd** does a significant amount of processing at connect time, it is more efficient to perform multiple commands in a single session than bto initiate a distinct connection and login for each command. The simplest way to have a single **nrcmd** session perform multiple commands is to create a batch file with one command per line and to redirect standard input from that file. A more complicated approach, but one that provides more control over the command sequence, is to run **nrcmd** from a controlling program and have that program send commands and read their status and output.

Using nrcmd Batch Files

The simplest way to automatically perform multiple configuration commands is to create a batch file of **nrcmd** commands and have them executed sequentially. For example, to create a scope and add reservations to it, you can enter these commands and store them in the file scope.txt. Lines beginning with the pound character (#) are comment lines:

```
# This set of commands creates a scope and adds four reservations
scope demo1 create 24.10.2.0 255.255.255.0
scope demo1 addReservation 24.10.2.1 1,6,0a:23:45:67:89:01
scope demo1 addReservation 24.10.2.2 1,6,0c:23:45:67:89:02
scope demo1 addReservation 24.10.2.3 1,6,0c:23:45:67:89:03
scope demo1 addReservation 24.10.2.4 1,6,0a:23:45:67:89:04
```

Note End the last command line with a newline character, or the command will not be executed.

You can then run a single **nrcmd** session to execute all of these commands:

```
% nrcmd -b < scope.txt
```

The advantage to using batch files is that you can execute multiple configuration commands while only incurring the connection cost once. However, if a command, such as the initial scope creation in the previous example, fails, the batch file continues even though subsequent commands are now useless.

You can use the *assert* function of the **session** command to perform simple logic checks. This command allows a **nrcmd** batch script to assert that a given condition is true. If the condition is true, the command has no effect; if false, the batch file is terminated at that point. For example, before executing the set of **scope** commands in the scope.txt file in the previous example, you might want to ensure that the cluster is locked:

```
# Quit if cluster cannot be locked
session assert locked
```

For details about the **session** command, see the "session" section on page 2-174.

Command Syntax

When you execute **nrcmd** commands that contain equal signs, you must put them within quotation marks. For example, to use a single command to create a client-class name, enter:

nrcmd -C *cluster* **-N** *name* **-P** *password* **"client** *MAC* **create client-class-name=***name***"**

Adding Program Control

A more sophisticated method for automatically configuring and controlling Network Registrar is to have a program or script start a **nrcmd** session and communicate with the session through standard input and output.

To control **nrcmd** from another program, you need to start **nrcmd** from the controlling program and redirect standard input and output from **nrcmd** to file handles in the controlling program. The controlling program can then write commands to the input file handle and read results from the output file handle.

When running in batch mode, **nrcmd** reads a line of input at a time and prints a new line after the prompt. This provides an easily parsed sequence of lines in response to any command where:

- Syntax is *status-line result-lines prompt-line*.
- *status-line* has the format [0-9]{3} .*.
- There may be zero or more *result-lines* of any format.
- *prompt-line* is **nrcmd**>.

The exact details of starting up **nrcmd** as a child process, and writing to and reading from its standard input and output, are specific to the programming language you use to implement the controlling program and are beyond the scope of this document.