CHAPTER 4

CLI Control Commands

Use the commands described in this chapter to monitor and control the CLI session. This chapter also describes the **ping** command, which you can use in combination with the **set** and **show** commands (described in the first two chapters of this book) to monitor and control a LightStream 2020 multiservice ATM switch (LS2020 switch) from the CLI.

Note Do not leave the CLI running idle. Exit the CLI when you have finished using it, or when you anticipate an extended period of inactivity in the CLI session.

Like any complex computer program, the CLI consumes some network processor (NP) or network management system (NMS) resources even when running idle. Exiting the CLI when you have finished helps optimize NP and NMS performance.

The CLI control commands are as follows:

clear	Clear the screen.
exit	Exit protected mode or CLI program (same as quit).
help	Provide online help for CLI commands.
password	Change the password for protected mode.
ping	Send ICMP echo packets to a host and report on any returned packets.
protected	Allow access to protected mode.
quit	Exit protected mode or CLI program (same as exit).
shell	Execute a LynxOS command under a copy of the LynxOS shell.
source	Execute a script of CLI commands.

clear

Use the **clear** command to clear the screen.

Syntax

```
clear
```

exit

Use the **exit** command to exit from protected mode or from the CLI program.

Syntax

```
exit
```

Description

If you are in normal mode, the **exit** command halts the CLI program and returns you to the bash prompt. If you are in protected mode, the **exit** command returns you to normal mode in the CLI, so that you must type **exit** a second time to halt the CLI.

Note The **exit** command provides the same functions as the **quit** command.

---

help

Use the *help* command to display online help for CLI commands.

Syntax

```
help [topic]
```

Argument

topic Specifies the name of the command on which you want help. The *topic* argument is optional. By default (if you do not enter a *topic* argument), the **help** command displays a list of all CLI commands.

Description

The **help** command displays online help information about CLI commands.

The *LightStream 2020 Network Operations Guide* describes the online help facility in the CLI.

Example

The following example shows the display that the **help** command displays with **protected** as its argument:

```
cli> help protected
NAME

protected - enter protected mode

SYNTAX

protected

DESCRIPTION

        Enter protected mode. User will be prompted for a password.
        Password will not be echoed as it is entered. Protected mode
        is required for sensitive or potentially disruptive operations.

cli>
```

password

Use the **password** command to change the password for the npadmin account, which is also the password for protected mode. The **password** command requires protected mode.

Syntax

```
password
```

Description

Use the **password** command to change the password for the npadmin account, which is also the password for accessing protected mode in the CLI.

Note There are four accounts on the NP: oper, npadmin, root, and fldsup. To change the password for any of these accounts, use the LynxOS **passwd** command. (Note the difference in spelling. The LynxOS **passwd** command is described in the *Lightstream 2020 Network Operations Guide*, in the chapter on the command-line interface.) There are thus two ways to change the npadmin password (which is also the password for protected mode), but only one way to change the root, oper, or fldsup password. The root password also admits you to protected mode.

The **password** command affects *only* the node on which the CLI is running when you execute it, regardless of any target set with the command **set snmp hostname** *name*. It takes effect when the CLI is restarted on the affected node.

Example

The following is an example of using the **password** command to change the npadmin password:

```
*cli> password
Changing NIS password for npadmin on Boston5.
Old password:
New password:
Retype new password:
NIS entry changed on Boston5
*cli>
```

ping

Use the **ping** command to send ICMP echo packets to a host and report on any returned packets.

Syntax

```
ping [packet-size] host-name
```

Arguments

packet-size Specifies the size of the packets that are sent. The *packet-size* argument is optional. The default packet size is 64 bytes.

hostname Specifies the name of the host to which to send packets. The *hostname* argument is an IP address specified either by name (for example, nnsc.nsf.net or boston5) or by number (for example, 197.5.0.8 or 198.113.178.17). See the *LightStream 2020 Network Operations Guide* for information about creating the hosts file that defines host names.

Description

The **ping** command sends a series of ICMP echo packets at 1-second intervals to the specified IP address and reports on any returned ICMP echo-response packets. Press **^C** (the control-C key combination) to stop the **ping** process and display a summary of the results.

Note The **ping** command can impose a significant load on the network. For that reason, it is unwise to use **ping** during normal operations or from automated scripts.

This command affects *only* the node on which the CLI is running when you execute it, regardless of a target set with the command **set snmp hostname** *name*.

Example

The following example shows four ping packets sent to the node named boston5:

```
cli> ping boston5
PING boston5 (198.113.178.17): 64 data bytes
64 bytes from 198.113.178.17: icmp_seq=0 time=26 ms
64 bytes from 198.113.178.17: icmp_seq=1 time=12 ms
64 bytes from 198.113.178.17: icmp_seq=2 time=13 ms
64 bytes from 198.113.178.17: icmp_seq=3 time=14 ms
^C
----boston5 PING Statistics----
4 packets transmitted, 4 packets received, 0 packet loss
round-trip (ms) min/avg/max = 12/17/26

cli>
```

The IP address 198.113.178.17 could have been used as the command argument instead of the alias boston5. The command used the default packet size of 64 bytes. The example shows the number of packets transmitted, the number received back from boston5, and the minimum, average, and maximum round-trip transmission times.

protected

Use the **protected** command to allow access to protected mode.

Syntax

```
protected
```

Description

The protected command is used to access the CLI in protected mode. When you are in protected mode, you have access to additional commands. The **protected** command prompts you for the password for protected mode. You cannot enter protected mode unless you enter the correct password. When you are operating in protected mode, the cli> prompt is preceded by an asterisk (*cli>). To terminate protected mode and return to the cli> prompt, use the exit or quit command.

Note The password for protected mode is the same as the password for the npadmin account. The root password also admits you to protected mode.

---

quit

Use the **quit** command to exit protected mode or the CLI program.

Syntax

```
quit
```

Description

The **quit** command halts the CLI program. If you are in protected mode when the **quit** command is issued, you exit from protected mode, but remain in the CLI (in normal mode), so that you must type **quit** a second time to exit from the CLI.

Note The **quit** command provides the same functions as the **exit** command.

shell

Use the **shell** command to execute LynxOS commands under a copy of the LynxOS shell. The **shell** command requires protected mode.

Syntax

```
shell "LynxOS command"
```

Arguments

LynxOS command The argument is any LynxOS command that you can normally execute from the bash shell. Place the command in quotation marks.

Description

The **shell** command executes a LynxOS command as if the command were being run from the bash shell. The actual LynxOS command must be surrounded by quotes.

For additional information about LynxOS commands, see the *LightStream 2020 NP O/S Reference Manual*. If more than one command is to be executed, you may find it more convenient to start a subshell with the command **shell bash** at the cli> prompt. If you do this, you must terminate the subshell afterward with the **exit** command, which is built in to the bash shell.

Note This command affects *only* the node on which the CLI is running when you execute it, regardless of a target set with the command **set snmp hostname** *name*.

Example

The following example shows use of the shell command to run a LynxOS command that displays the date and time as they are known to the system. Because the command is a single word, the quotation marks are optional:

```
*cli> shell date
Thu Feb 17 13:39:52 EST 1994
*cli>
```

The following example shows use of the **ls** command to display names of all files that begin with a dot (.) plus an `r`. Because the command contains a space, it must be entered within quotation marks:

```
*cli> shell "ls .r*"
.rhosts       .rhosts.dist
*cli>
```

The metacharacter **?** can be used in the quoted argument string of the **shell** command only awkwardly. For example, the shell wildcard sequence **.??*** matches all file names that begin with a dot plus at least two other characters; you need two **?** characters here because typing **.*** or **.?*** would

include **..** (the abbreviation for the parent directory). However, the CLI attempts to interpret **?** as a request for help, before you can type the closing quotation mark, and then complains that the quotation mark is missing. This is illustrated by the following example:

```
*cli> shell ”ls .?
missing quote
*cli>
```

As a workaround, type **?** and ignore the error message, then type **^P** to repaint the command and continue typing. The following example demonstrates this workaround.

```
*cli> shell ”ls .?
missing quote
*cli> shell ”ls .??
missing quote
*cli> shell ”ls .??*”
.bash_history  .profile        .rhosts           .rhosts.dist
*cli>
```

Note that the error message is evoked twice in the example, once for each time **?** is typed. Although the printed example cannot show this, after each appearance of the error message, **^P** is typed to repaint the previous command.

The following example shows use of the **cbufpr** command to display the last lines of the mma.traplog file. The command contains spaces, so it must be enclosed in quotation marks:

```
*cli> shell ”cbufpr -t /usr/tmp/mma/mma.traplog”
PROGRAM: cbufpr: compiled Feb 02 1994 @ 16:07:57 [pid:35]
(OPER) NDD_3 at 02/17/94 16:35:29 EST (02/17/94 21:35:29 GMT)
       Line Card lstb4:3 (MS-TR) up.
(INFO) NPTMM_2007 at 02/17/94 16:35:30 EST (02/17/94 21:35:30 GMT)
       Slot 3 State Changed From DOWN To UP
(INFO) LC_2000 at 02/17/94 16:35:30 EST (02/17/94 21:35:30 GMT)
       Slot 3: ND: ERMP channel to NP open
.
.
.
(INFO) LC_2000 at 02/17/94 16:35:40 EST (02/17/94 21:35:40 GMT)
       Slot 8: ports enabled 0x00
(OPER) NDD_3 at 02/17/94 16:35:46 EST (02/17/94 21:35:46 GMT)
       Line Card lstb4:6 (LS-EDGE) up.
*cli>
```

Use the command **bash** to start a subshell. It is necessary to do this for executing multiple commands, and it makes it easier to execute a command containing the metacharacter **?** (described with an earlier example). The following example illustrates use of the **cd** command to change to the directory containing the cli.groups file, then the **vi** editor to edit that file, and finally the shell built-in command **exit** to return to the CLI:

```
*cli> shell bash
LSnode:2$ cd /usr/app/base/config
LSnode:2$ vi cli.groups
.
.
.
```

(The displays from the vi editing session are omitted from this example.)

```
.
.
.
LSnode:2$ exit
exit
*cli>
```

The effects of the **cd** command are lost when you terminate the subshell. The next use of the **shell** command executes LynxOS commands in your login directory.

source

Use the **source** command to execute a script of CLI commands stored in a disk file.

Syntax

```
source "scriptname"
```

Argument

"scriptname"	Identifies the file that contains the CLI commands that you want to execute. If the file *scriptname* is not in the current directory (usually your login directory), you must enter the full pathname of the file.

Note You must surround the file name or pathname with quotation marks.

Description

The *source* command executes a script of CLI commands. A script is a logical sequence of commands stored in a disk file. The file does not have to be executable (it is not interpreted by the shell). Before executing each command, the CLI prints a **+** character as a prompt and echoes the command line as it appears in the script file.

The simplest way to create a CLI script is to use the **echo** command at the bash prompt (see the following example). You can also use the vi editor to create the script. Each command must begin on a new line. Comments can be included with the CLI commands in C programming style, between an initial **/*** string and a final ***/** string.

You can interrupt extended output of a command in the script by typing **^C**. The software drops buffered output and goes on to the next command.

Note This command affects *only* the node on which the CLI is running when you execute it, regardless of a target set with the command **set snmp hostname** *name*.

Example

The following example shows how to use the **echo** command to create a file containing multiple commands:

```
LSnode:2# echo ”show chassis cards show card 2 ports” > card3.script
```

The following example shows how to use the **cat** command to verify the contents of the new script file:

```
LSnode:2# cat card3.script
show chassis cards
show card 2 ports
LSnode:2#
```

The following example shows the output that results when the example script file is executed with the **source** command:

```
cli> source ”scriptfile”
+ show chassis cards
Slot 1: NP
Slot 2:  OC3 Trunk
Slot 3:  OC3 Edge
Slot 4:  MS Trunk
Slot 5:  MS Trunk
Slot 6:  MS Trunk
Slot 7:  LS Edge
Slot 8:  OC3 Trunk
Slot 9:  FDDI
Slot 10: Ethernet
Slot SA: Switch2
Slot SB: Switch2
+ show card 2 ports
Port  2000 CLC Trunk    Name: tb5.2.0
cli>
```