234_nokia_08.qxd 3/3/03 3:04 PM Page 241

Chapter 8

Advanced System Administration

Solutions in this chapter:

- **Understanding the Boot Manager**
- **Using CLISH**
- **Troubleshooting**

☑ **Summary**

☑ **Solutions Fast Track**

☑ **Frequently Asked Questions**

234_nokia_08.qxd 3/3/03 3:04 PM Page 242

Introduction

Despite the fact that Nokia has worked hard (and has mostly succeeded) at making Voyager the focal point of system configuration and monitoring, at times it is not adequate, and sometimes when power users want to stray from its grip and drop into the more familiar and powerful command shell. IPSO 3.6 has made this switch much easier with the addition of *CLISH*, the new Nokia command-line configuration shell.

Let's talk first about the IPSO boot sequence and how to use the boot manager. When things really go badly, we talk about how you can use system debugging tools to assist your interaction with Nokia support.

The IPSO command shell is simply the UNIX C-shell, or csh, so it is possible to write shell scripts using the shell built-in commands and system binaries. The shell and system binaries provided are based on those from FreeBSD 2.2.6, so if you need a command reference, you can use those UNIX manual pages, which you can find on the Internet. A good, useful subset of the FreeBSD base installation is part of the IPSO operating system. One section is devoted entirely to using the CLISH shell, new in IPSO 3.6.

Using the shell means you have access to all your familiar UNIX commands: *vmstat*, *netstat*, *ifconfig*, *tail*, *grep*, *df*, and many others. Various other useful utilities, such as Perl and the Bash shell, have been ported to IPSO. (See Appendix B for details on getting and installing these packages.) We will talk about some of the most useful commands for troubleshooting and show you when you might want to use them. In those rare cases where we must use Voyager, we will stick to the lynx interface, to remain as close as possible to the command line.

Understanding the Boot Manager

The Nokia *boot manager* is a small program that runs just after system startup but before the operating system kernel is loaded into memory. Its main function is to load the kernel from disk into memory, which then handles normal system startup and initialization. Nokia's boot manger has gone through several changes over the years and has been present on the system hard drive, a specially formatted diskette drive, or (most recently) in flash memory, the latter to ease upgrades and provide some measure of resiliency in the event of a hard disk crash. The boot manager will, if left unattended, simply bootstrap the system with the default kernel image, but it can be interrupted and given options from a rudimentary command shell. This functionality is typically useful, for example, to boot into "single-user" or nonnetworked mode for system maintenance.

234_nokia_08.qxd 3/3/03 3:04 PM Page 243

The boot manager in the IP300, IP500, and higher series allows you to do various things that would be impossible if your Nokia device were already booted fully. Chief among these tasks are the following:

- You can perform a factory default installation.
- You can boot from an alternate kernel and/or an alternate device.

The boot manager in the IP400 series of Nokia appliances is limited to allowing you to perform a factory default install, and then only from a boot diskette.

Single-User Mode

Before we can begin discussing the boot manager, you need to understand a system mode called *single-user mode*. This mode of operation is used frequently for performing system maintenance (such as upgrading the boot manager, for example) and is considered "safer" for such actions because it is nonnetworked and has fewer services running. The normal mode of operation of your Nokia appliance is called *multiuser* or *networked* mode. Your Nokia will boot into multiuser mode by default if it is allowed to boot unattended. Single-user mode must be forced at the boot manager prompt. The procedure for entering single-user mode differs depending on the model of Nokia you are using.

The IP440

To get your IP400 series appliance into single-user mode, reboot or power-cycle it from a console connection. When you see the boot: prompt, you must enter **-s** and press **Enter** within 10 seconds or the device will boot into its normal multiuser mode of operation. Once the boot process into single-user mode is complete, you will be asked to **Enter pathname of shell or RETURN for sh:**. Just press the **Enter** key to get a command shell. To exit, simply type **exit** or press **Ctrl + D** (hold down the Control key as you press the **D** key). This key combination causes you to leave single-user mode and restart the system.

The IP300 and IP500 or Higher Series

On an IP500 or higher series device, you will see the text *Entering autoboot mode. Type any character to enter command mode* during the startup sequence, at which point you can press any key before the 5-second timeout to get to the boot manager prompt. To get into the boot manager on an IP300 series device, wait until you see the *Verifying DMI Pool Data* line in the boot sequence, then press the

234_nokia_08.qxd 3/3/03 3:04 PM Page 244

number **1** to start loading boot manager. At this point, the procedure is identical to the IP500 series: Just wait until you see *Type any character to enter command mode*, then press any key before the 5-second timeout to get to the BOOTMGR[0]> prompt. Now type **boot –s** and press **Enter**. Again, after the system boots, it will ask you to *Enter pathname of shell or RETURN for sh:.* Press the **Enter** key to get a command shell. To exit single-user mode, type **exit** or press **Ctrl + D**. Either choices causes you to leave single-user mode and restarts the system.

In both cases, pressing any key before the configured timeout period at the boot sequence prompt will stop the timer and keep you in the boot manager until you exit or resume the boot sequence manually. We discuss booting manually in the section "Boot Manager Commands" later in the chapter.

NOTE

It is also possible to enter single user mode from a running system. Type **kill –TERM 1** (you must be logged in as admin on a console for this to work) to send the *init* process a TERM signal and force entry into single-user mode. The same thing can be accomplished by typing **shutdown now** from a console login.

Understanding the IPSO /image Directory

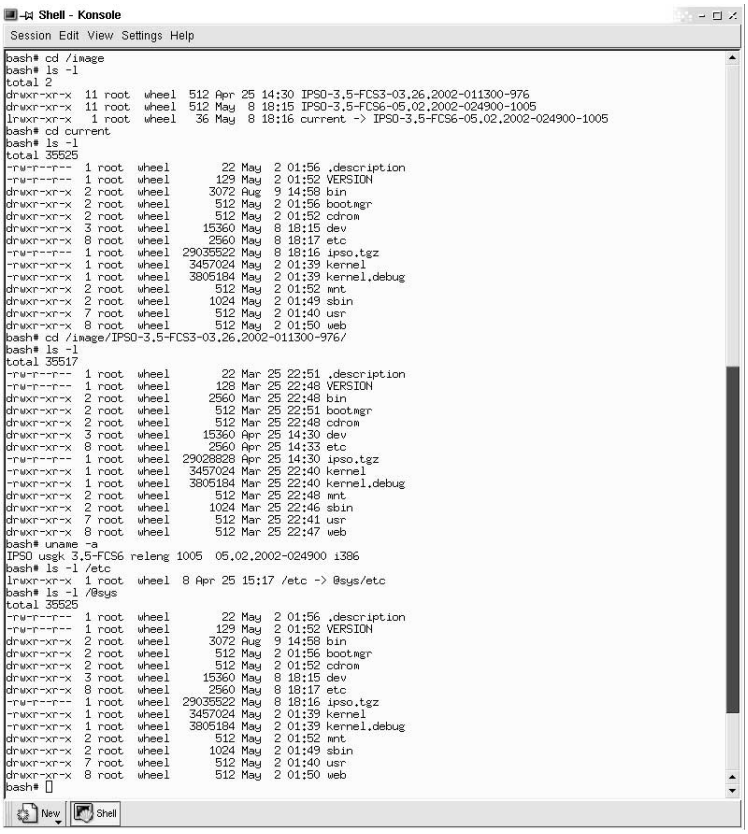
Nokia has devised a clever way of saving old IPSO images and file systems and making them available after upgrades in the event a downgrade is ever necessary. As we discussed in Chapter 2, the file /image/current/kernel points to the kernel image that is booted by default when your Nokia system is left to start unattended. In practice, there is more going on under the hood than merely a symbolic link to a kernel image file.

Let's take a look at the directory structure of /image in detail. In Figure 8.1, we see that /image/current is actually a symbolic link to the directory IPSO-3.5-FCS6-05.02.2002-024900-1005, which itself contains almost an entire IPSO file system tree. Executing the *uname –a* command shows us that this is indeed a Nokia with IPSO 3.5 FCS6 installed and running. Executing a **cd /image/IPSO-3.5-FCS3-03.26.2002-011300-976** command places us in the previous IPSO's file system tree, so we can see our previous kernel image file as well as the /bin, /usr, /sbin, and /etc directories of the old IPSO. So where do the current

234_nokia_08.qxd 3/3/03 3:04 PM Page 245

system's /bin, /usr, /sbin, and /etc directories point? To the corresponding directories in the /image/current directory, of course. If you execute an *ls –l /etc* command, we see that /etc is merely a symbolic link to /@sys/etc. We would see a similar pattern for /bin, /usr, /sbin, /web, and /dev. @sys is merely Nokia's link to whatever directory /image/current points to. To see this, type **ls –l /@sys**, which shows exactly the same output as when we executed *ls –l* from the /image/current directory.

Figure 8.1 A Trip Through the IPSO /image Directory



Boot Manager Variables

Now we have enough background information to talk about how IPSO's boot manager works. The boot manager operates by referencing some user-definable variables that are given sensible defaults when your Nokia device is shipped. Table 8.1 lists each variable and its purpose. This section discusses how to change the value of these variables.

234_nokia_08.qxd 3/3/03 3:04 PM Page 246

Table 8.1 User-Definable Boot Manager Variables

Variable Name Value	Meaning	Factory Default
autoboot	Do we wait for *bootwait* seconds at the boot manager prompt during startup before continuing unattended?	Yes
boot-device	Device to load the boot-file from	wd0
boot-file	Kernel image path	/image/current/ kernel
boot-flags	Flags to pass to the kernel	-x
-x	Do not identify the flash disk as wd0	N/A
-d	Enter the kernel debugger as soon as possible during startup	N/A
-s	Single-user mode; admin password may be needed if console is marked "insecure" in /etc/ttys	N/A
-v	Verbose mode	N/A
bootwait	The amount of time to wait at the boot manager prompt for user input before continuing	5 seconds

NOTE

If you set autoboot to *No*, your Nokia device will stop at the boot manager prompt during startup and wait indefinitely for your keyboard input. In this case, you would need to manually enter the *boot* command from a console connection to get the system up and running.

Boot Manager Commands

Once you are at the boot manager prompt, you can enter various commands. Here is a list of those commands with sample uses:

- **printenv** Prints all variables and their values to the screen.
- **showalias** Shows alias list in volatile memory. Eight aliases are available.

234_nokia_08.qxd 3/3/03 3:04 PM Page 247

- **sysinfo** Shows CPU, memory, and device information.
- **ls** The syntax of this command is ls *device path*; this will view the contents of a directory given by *path* on the device *device*. For example, *ls wd0 /image/current* will list the contents of the currently active IPSO directory tree.
- **setenv** This command is used to set environment variables. The syntax you should use is setenv *name value*. For example, to change the default 5-second boot timeout to 10 seconds, enter **setenv bootwait 10**.
- **unsetenv** This command unsets or clears the environment variable given to it by name. So, for example, *unsetenv boot-file* clears the boot-file variable. Note that unsetting autoboot will not clear it but will set it to *No*. Similarly, unsetting *bootwait* will set that variable to 0.
- **set-defaults** Resets the given environment variable back to its default value, so *set-defaults bootwait* would set bootwait back to 5 seconds. If you use set-defaults with no arguments, the system resets all the boot manager environment variables to their default values.
- **setalias** Used to set an alias. An alias allows you to substitute one name for another. The syntax for this command is *setalias <name device>*, where *name* is the alias you want to create, and *device* is the name of the device you want to alias. Note that *disk* is a predefined alias for *wd0*.
- **showalias** Displays a list of all the currently defined aliases.
- **unsetalias** Used to delete or undefine an alias. The syntax for this command is *unsetalias <name>*, so the command *unsetalias disk* deletes the alias named *disk* that you defined earlier.
- **halt** Used to halt the system, which is typically followed by the system being powered off. Note that *halt* can also be used from a multiuser or single-user shell and is the safest way to power off your Nokia device, since it makes sure that all mounted file systems are unmounted.
- **help** Show help for the various boot manager commands.
- **boot** The *boot* command is used to boot the system manually. It allows you to specify that you would like to boot from a specific device, with a specific kernel image, using explicit kernel flags. It is sometimes useful to restore a system that has been rendered unbootable by a failed upgrade or corrupt kernel image. The syntax of the boot command is

234_nokia_08.qxd 3/3/03 3:04 PM Page 248

boot <boot-device> <boot-file> <boot-flags>. For example, typing **boot wd0 /image/current/kernel.old** would boot the kernel named *kernel.old* in the /image/current directory on device *wd0*. Typing just the word **boot** here has the same effect as if you had typed **boot wd0 /image/current/kernel** (/image/current/kernel always points to the kernel image of your most recently installed IPSO). Typing **boot –s** boots your Nokia into single-user mode.

- **install** Performs a factory default installation. This command is discussed later in the section.
- **passwd** Normally, access to the *install* command in the boot manager is open to anyone with physical access to your Nokia appliance; no password is necessary. If you are not confident in the physical security measures at your workplace, it is probably a good idea to set a password for access to this command. The first time you use *passwd*, it will prompt you for a new password twice. Once the *install* password has been set, you will be prompted for the current password before being allowed to change it again.

WARNING

Be careful! If you lose or forget the *install* password, in order to get access to it again you will have to remove your hard drive(s), reboot your Nokia appliance, change the install password, reinstall your hard drive(s), and reboot again. Some older appliances without the boot manager in flash memory must be returned to Nokia to have the password reset.

Reinstalling or Upgrading the Boot Manager

Reinstalling or upgrading your boot manager is sometimes necessary. Reinstalling your boot manager is usually something your support provider will ask you to do in rare circumstances—perhaps to rescue a system after a failed IPSO upgrade. Manual upgrade of your boot manager is needed only when you upgrade IPSO from version 3.2.1 to version 3.4 and later. Starting in IPSO version 3.3, the boot manager is upgraded automatically for you if needed, so we recommend upgrading to version 3.3 first, before going to any later version. You must be in single-user mode to reinstall or upgrade your boot manager. The procedure is outlined here:

234_nokia_08.qxd 3/3/03 3:04 PM Page 249

1. Download the boot manager specific to your hardware model and IPSO version on a workstation or laptop that has network connectivity to your Nokia appliance. To find the correct boot manager image file, go to http://support.nokia.com, log in, and follow the Release Notes and Software Download link for the version of IPSO you are interested in. The link to the boot manager image is under the heading Boot Manager Upgrade Images for the IP300 or IP500 and later series and under the heading Boot Floppies for the IP400 series. Note that you do not have to upgrade the boot diskette on an IP400 series device sunless you want to perform a factory-default install.
2. Transfer the image file you just downloaded to your appliance using FTP or SCP, remounting your root partition read/write with the command **mount –uw /**.
3. Now log into your Nokia, and check the md5 hash of the downloaded image with **md5 <*path to downloaded image file*>**. The correct hash is displayed on Nokia's Web site, just below the image download link.
4. For the IP300 and IP500 or higher series, copy the downloaded image file to /etc/nkipflash.bin.
5. For the IP400 series, copy the image to a diskette using the UNIX **dd** command. On your Nokia device itself, you would place a diskette in the drive and run the command **dd if=< *image path*> of=/dev/rfd0**.
6. Type **reboot** and follow the preceding instructions for getting your device into single-user mode.
7. Since the IP400 series of devices uses the boot diskette as its boot manager, you are done. The boot diskette will automatically start the factory-default install process. (See the next section for details.)
8. If you have an IP300 series device, use the following commands to install or upgrade your boot manager:
   - Install: **/etc/install_bootmgr wd0 /etc/nkipflash**
   - Upgrade: **/etc/upgrade_bootmgr wd0 /etc/nkipflash**
9. If you have an IP600 or IP700 series model, use the following commands:
   - Install: **/etc/install_bootmgr wd1 /etc/nkipflash**
   - Upgrade: **/etc/upgrade_bootmgr wd1 /etc/nkipflash**

234_nokia_08.qxd 3/3/03 3:04 PM Page 250

10. Reboot with the command **reboot**. Your Nokia device should stop at the boot manager prompt just after it restarts, because all its boot manager environment was wiped clean by the upgrade or install. Simply type **set-defaults** at the boot manager prompt to get them back to their default values.
11. Proceed with your IPSO upgrade using the boot manager's **install** command or use **newimage** from a shell prompt. If you were just reinstalling a previous boot manager version, bring up the system manually with the boot manager's **boot** command. (See the next section for the specifics of the boot command.)

Performing a Factory-Default Installation

You might find it necessary to reinstall IPSO as thought it just came from the factory—perhaps if you are configuring a previously used Nokia device or have a damaged installation. As discussed earlier, the boot manager's *install* command (or the boot diskette on an IP400 series model) accomplishes this task. This procedure will reformat and repartition your hard drive, so make sure that you have performed appropriate backups before starting. The procedure is as follows:

1. Set up an FTP server and make sure the Nokia appliance you are reinstalling has physical network connectivity to it. Test your FTP server from another host to make sure it works. The IP400 series has the capability of pulling the required files from its CD-ROM drive instead of from an FTP server.
2. Download the necessary IPSO package (ipso.tgz) and whatever other packages you want (such as FireWall-1) from either Nokia or Check Point's support site. Place the packages on your FTP server or burn them onto a CD (IP400 series only).
3. Get to the boot manager prompt as described in the "Single-User Mode" section. On an IP400 series device, you have to boot from either the boot diskette provided with your appliance when it was shipped or create a new boot diskette with the correct boot manager for the version of IPSO you are reinstalling. *The boot diskette will automatically start the reinstall procedure when you boot with it in the drive, so skip to Step 7 at this point.*
4. Use the boot manager's *printenv* command to see the version of your boot manager, and make sure it is compatible with the version of IPSO

234_nokia_08.qxd 3/3/03 3:04 PM Page 251

you are installing. Use the guidelines in the section "Reinstalling or Upgrading the Boot Manager," if necessary.

5. Type **install** and press **Enter** from the boot manager prompt.
6. Enter the install password, if you set it up previously.
7. Follow the prompts, which will have you enter your device serial number, specify both your IP address and the IP address of your FTP server, and specify the username and login for your FTP server. You have the option of getting the required files from a CD-ROM drive on the IP400 series.
8. Once IPSO has been transferred and installed on your Nokia, reboot with the **reboot** command.
9. After a reboot, you will be back at the Hostname? prompt, as if this were a freshly unpacked box. Refer to Chapter 3 for the new install procedure.

Using CLISH

Command-line configuration before IPSO 3.6 was always a very difficult proposition on Nokia firewalls, given the way IPSO handled system configuration. As we discussed in Chapters 2 and 4, any changes you make to configuration files in /etc are lost at the next reboot, and changes made with standard UNIX tools such as ifconfig might not last even 5 minutes due to system daemons such as *ifm* and *pm* that monitor interfaces or critical processes and restart or reconfigure them on the fly, according to the settings in /config/active. The usual way to make system configuration changes was to edit the /config/active file, then synchronize it with the relevant configuration file in /etc with the **_xlate* series of commands. This method was imperfect, however, because anyone starting up Voyager would not see your changes and could override them accidentally unless a reboot was done after every change. The *dbset* command could also be used to make changes from the command line, but very few of the *dbset* command sequences were documented by Nokia, at least for public consumption.

User demand and the need for a scriptable interface that could enable remote configuration changes caused Nokia to release the *Command Line Interface Shell*, or *CLISH*, starting in IPSO 3.6. CLISH provides a simple command-line interface to *all* Voyager configuration options, either through a shell-like environment or from a file containing CLISH command sequences. If you are familiar with the secure, remote command execution capabilities of SSH, you will see the

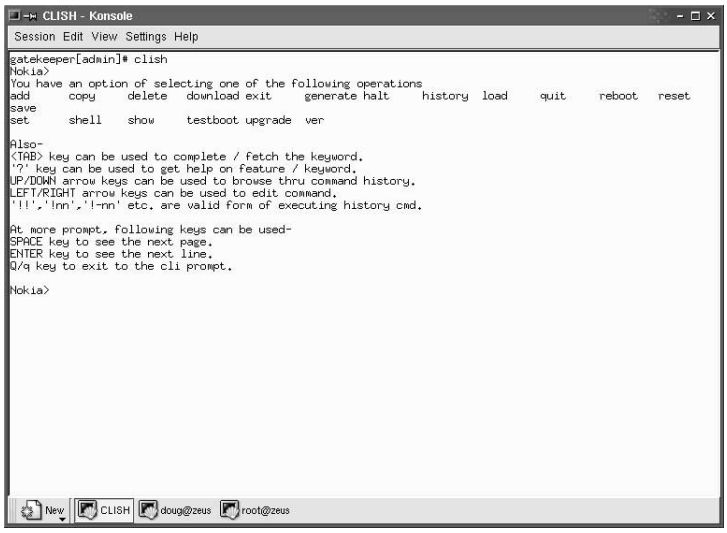
234_nokia_08.qxd 3/3/03 3:04 PM Page 252

potential that exists for a remote Nokia management tool. In this section, we talk about how to enter and use the CLISH shell interface to make permanent system changes, without going near Voyager.

CLISH Basics

To enter CLISH, type **clish** and press **Enter** from a command prompt. You will see a Nokia> prompt, ready to accept commands. Although CLISH comes with a large reference guide, it really is not necessary for most commands. Simply press the Help (**?**) key at the CLISH prompt and a basic help screen will appear (see Figure 8.2).

Figure 8.2 The CLISH Help Screen

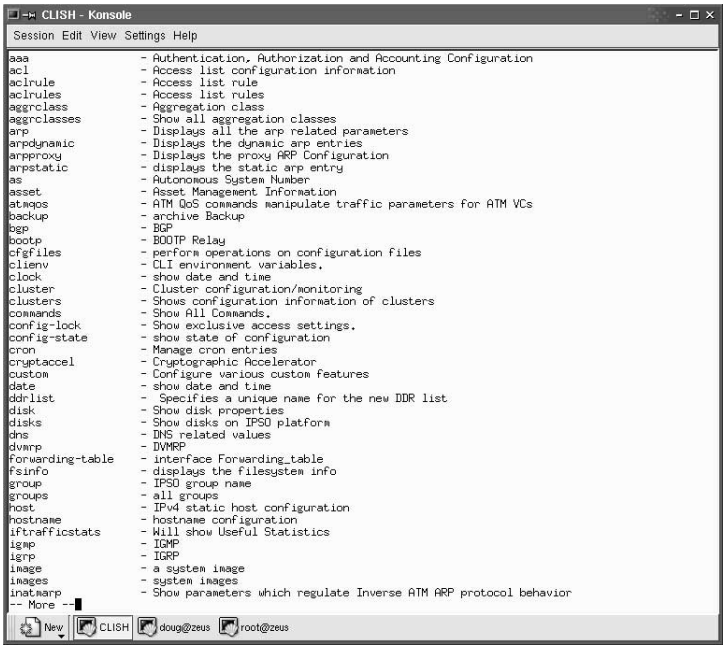


The help operator (?) can even be used in the middle of a command string, to give context-sensitive help on the argument you just typed and what needs to follow. Some of the other nice features CLISH offers are Tab completion of commands, command history perusal with the Up and Down Arrow keys (both of which work much like the Bash shell), basic command-line editing with the Right and Left Arrow keys, and C-shell style history search with the ! operator. Pressing **Ctrl + u** will erase the current line; pressing **Ctrl + c** will abort the current command. **Quit** or **exit** causes you to leave CLISH, although this step is unnecessary if you just want to execute a shell command or two. In that case, type the **shell** command to spawn a subshell directly from CLISH; when you are done entering commands, type **exit** to get back to the CLISH prompt.

234_nokia_08.qxd 3/3/03 3:04 PM Page 253

If you want to see what commands are available from one of the main sections shown on the basic help screen, type that command and press **Tab**. You will see a list of valid command arguments that can follow whatever you just typed. For example, if you want to see what the *show* command has to offer, type **show** and press **Tab**. You will see something like the screen shown in Figure 8.3. (Notice the *More* reminder at the bottom of the screen; it indicates there are more completion choices, which you can access by pressing the **Spacebar**.)

Figure 8.3 Show Command Completions in CLISH



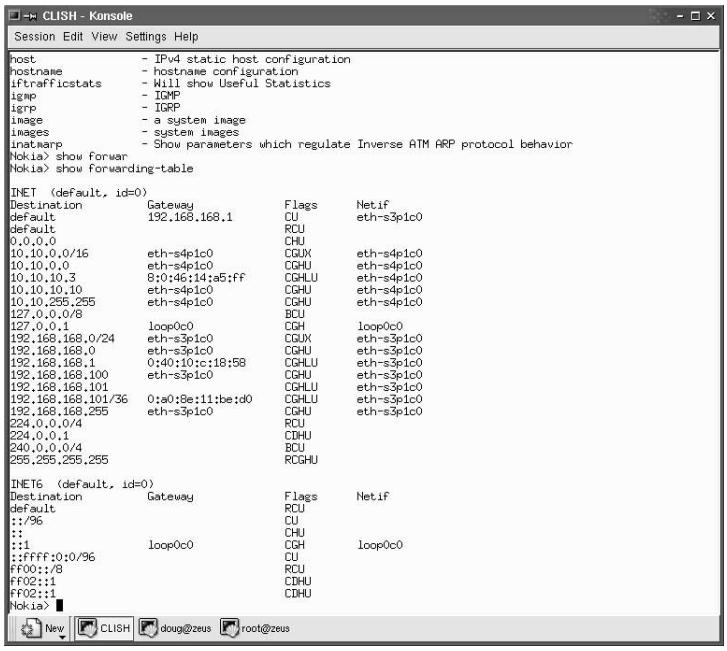
If you press **q**, you are put back at the *show* command you just typed, with a trailing space added for whatever you want to type next. You only ever need to type just enough of the command or argument for it to be unique before pressing the **Tab** key again, so type **fowar**, press **TAB**, and then press **Enter**. When you press Tab, the phrase *forwarding-table* is completed for you and is then displayed (see Figure 8.4). To get this command with the least amount of typing, you could have simply typed **sho** and pressed **Tab**, then typed **fo** and pressed **Tab**, then pressed **Enter**.

CLISH also has the capability to execute just one command or multiple commands from a file in *batch mode*. To execute just one command, type **clish –c "command"** from a shell prompt. To execute more than one command, place the

234_nokia_08.qxd 3/3/03 3:04 PM Page 254

commands (one per line) in a text file and execute them by typing **clish –f commandfile.txt**. The same file can be loaded and executed from within CLISH as well. From a CLISH prompt, type **load commands commandfile.txt**.

Figure 8.4 CLISH Display of the Forwarding Table



Commands entered via CLISH are applied to the running system immediately, but they are not persistent across reboots. Any configuration changes you make must be *saved* to be permanent—much like the Voyager apply/save semantics. To permanently save all your configuration changes from within the interactive shell, type **save config**. To save your configuration changes to a separate file named *configfile.txt*, type **save cfgfile configfile.txt**. This will be a full-blown valid system configuration file and is saved in /config/db with all the other system configuration files. Beware that the new configuration file is linked to /config/active once it is saved, effectively altering your system's active configuration. This might not be immediately obvious and is not documented. If you are using the one-off command form or the batch file form of CLISH, add the switch *–s* to the command to force a configuration save after the command or batch file executes, as in **clish –s –f commandfile.txt** or **clish –s –c "set vrrp accept-connections on"**.

In general, CLISH commands are formatted as *operator feature argument(s)*, where *operator* is usually one of *show*, *set*, *add*, or *delete*. Yes or no arguments are

234_nokia_08.qxd 3/3/03 3:04 PM Page 255

generally specified *yes* or *no* on the command line, similarly for *on* or *off*. If you know a feature name, using the word *default* for the argument always sets the system default value, even if you don't know what it is. This can be a good way to get your system back to normal if you enter a strange setting by mistake.

If you maintain a Nokia system for which there are more administrators besides yourself, it is helpful to block other administrators' ability to make configuration file changes. To do this, enter CLISH and type **set config-lock on**. Make your changes, then type **set config-lock off** to release the lock. You will get an error message if you try to lock the configuration when someone has already locked it.

Troubleshooting

When things go wrong, it is nice to know that you have a fully functional operating system under the hood that can help you get enough information to solve problems yourself or at least to help your support provider fix the problem. This section discusses the various command-line utilities that IPSO provides for performing things such as recovering lost passwords, log management and searching, packet sniffing, and memory usage collection. Many of these commands are merely simple UNIX commands that we use together in the way that they were intended, and some are creations of Nokia that allow you to gather information for them, should the need arise.

Managing Logs

The IPSO operating system provides several command-line utilities and many log files that can be used to troubleshoot problems or monitor user or process behavior. Most of the log files created and maintained by IPSO are in the /var/log directory, although some are in other subdirectories of /var. Check Point FireWall-1 logs are kept in /var/fw/log. We talk about some of the most useful utilities and log files in the following sections.

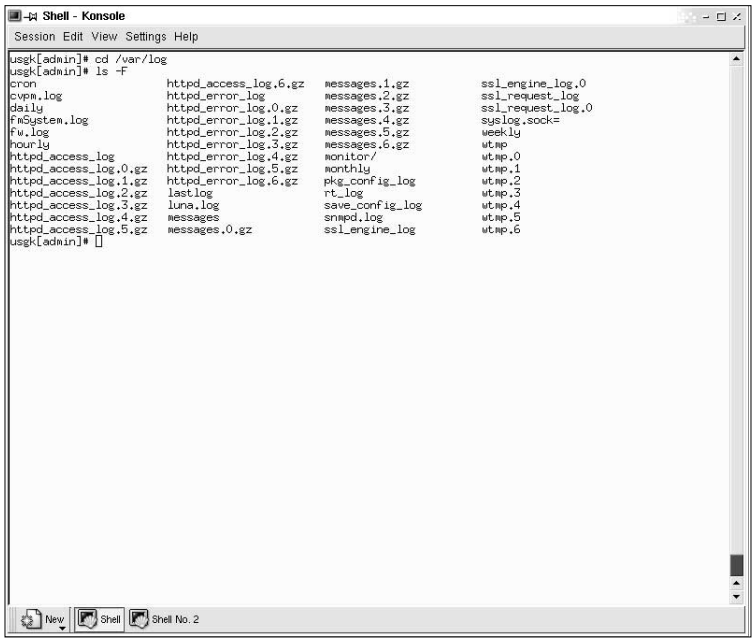
Searching and Displaying Log Files

Figure 8.5 shows the contents of the /var/log directory. The most useful file here is the one named *messages*. This is the global system log, sometimes called the *syslog*, and is preconfigured for you through settings in /etc/syslog.conf. Notice that, in addition to a messages file (which is text and can be displayed as such), there are seven files, named messages.0.gz through messages.6.gz. These are the compressed system logs for the past seven months. Every month, a system cron

234_nokia_08.qxd 3/3/03 3:04 PM Page 256

job runs that truncates the /var/log/messages file after compressing and saving it to /var/log/messages.0.gz. The old messages.0.gz file becomes messages.1.gz, and so on, until messages.6.gz is deleted and replaced by messages.5.gz. So IPSO will keep seven months' worth of archived logs for you while keeping your logs from growing uncontrollably and using up all the space in the /var partition. This process is known as *log rotation*. IPSO rotates some other logs in /var/log as well, including the HTTP access and error logs and the wtmp log that contains user login and logout times. Some log files are rotated more frequently; the system audit logs, for example, are rotated once per day because they have a tendency to grow in size very quickly. This is another nice feature of IPSO; not all UNIX operating systems do this for you.

Figure 8.5 Listing of the /var/log Directory



You can use the *more* command to page through a text file one screen at a time with the Spacebar, and you can combine *gzip* and *more* to display the compressed logs, without permanently decompressing the gzipped archive. *Tail* is useful to watch logs in real time, and *grep* is a powerful search tool. For firewall logs, the Check Point *fw* command can be quite useful and is typically much faster and more responsive than Check Point's log viewer GUI. The following are some basic examples of using these commands; refer to the online man pages at

234_nokia_08.qxd 3/3/03 3:04 PM Page 257

www.freebsd.org/cgi/man.cgi?manpath=FreeBSD+2.2.6-RELEASE for detailed information on these and many other IPSO commands. Documentation for Check Point's *fw* command can be found in the *FireWall-1 Reference Guide* available at www.checkpoint.com/support/downloads/docs/firewall1/4_1/CPRef.pdf (support login required):

- **more /var/log/messages** Pages through /var/log/messages one screen at a time.
- **gzip –cd /var/log/messages.6.gz | more** Decompresses /var/log/messages.6.gz and sends the output through a pipe to *more*.
- **tail –f /var/log/httpd_access_log** Watches the HTTP access log in real time.
- **grep –E '\<10\.100\.6\.1\>' /var/log/messages** Displays all the lines in /var/log/messages that contain the IP address 10.100.6.1. (The \< and \> character sequences are necessary so you don't match IP addresses such as 110.100.6.11).
- **grep –E –v 'LOG_INFO|LOG_NOTICE' /var/log/messages | more** Prints all the lines in /var/log/messages that do *not* contain the strings LOG_INFO or LOG_NOTICE and pages through the output.
- **grep 'LOG_CRIT' /var/log/messages | grep –E –v 'FW-1|ex_expire' | more** Displays all the lines in /var/log/messages that contain the string LOG_CRIT but do *not* contain the strings FW-1 or ex_expire and pages through the output.
- **fw log –nft** Displays the firewall logs in real time.
- **fw log –nft | grep –i 'icmp' | grep –i 'drop'** Displays in real time firewall log entries for the service icmp that are dropped by the firewall. (The *-i* tells the system to do a case-insensitive string match.)
- **fw log –nft | grep –E '\<10\.100\.6.\1\>' | grep –v 'domain'** Shows all real-time firewall log entries containing the IP address 10.100.6.1 but that are *not* DNS traffic.

NOTE

Remember that under normal firewall operating conditions, your firewall logs are sent to your management console (in a distributed Check Point installation), which in many networks is a Windows NT or 2000 server,

234_nokia_08.qxd 3/3/03 3:04 PM Page 258

not your Nokia box. The *fw log* commands will not display anything on a firewall module, since there are no logs to display. The *fw log* commands work on Windows NT or 2000, but the *grep* portion of those commands does not. We recommend installing the excellent free software package Cygwin (see www.gnu.org/philosophy/free-sw.html for what the term *free software* means) on your Windows management station in that case, because it will provide you with a Bash shell and all the familiar UNIX text-processing utilities, including *more*, *tail*, and *grep*. The *fw log* command is much less useful without these utilities, especially in high-traffic environments. You can get a Cygwin net installer from www.cygwin.com/setup.exe. The Cygwin package even includes the latest OpenSSH server and clients, which can greatly ease remote administration of a Windows server.

Automating SCP Log Transfers

Sometimes it is necessary to transfer log files to another host because either your /var partition is out of space or you want to get a log file to another workstation for detailed analysis of some kind. It is possible to use FTP to transfer log files from your Nokia device, but this process suffers from several problems:

- The transfers are in plain text and can be sniffed, including usernames and passwords.
- Automating FTP is more involved than automating SCP transfers.
- You must remember to use ASCII or binary mode to transfer files to a remote host, depending on the file type (for example, uncompressed text versus compressed binary files).

SSH, specifically SCP, addresses all these problems. It provides for secure, encrypted file transfers using RSA or DSA authentication. Once the procedure has been automated, it is a simple matter to schedule transfers using cron. Here are the steps you need to take to set up this type of authentication.

First, you must generate public and private RSA keys for the user who will be transferring the log files from the Nokia device. We assume that this is user admin, since this will be the most common case, although you only need an unprivileged user who has read access to the required log files at a minimum. Note that we use CLISH and lynx to generate the appropriate SSH keys; you could use the *ssh-keygen* command to generate a public/private key pair from the command line,

234_nokia_08.qxd 3/3/03 3:04 PM Page 259

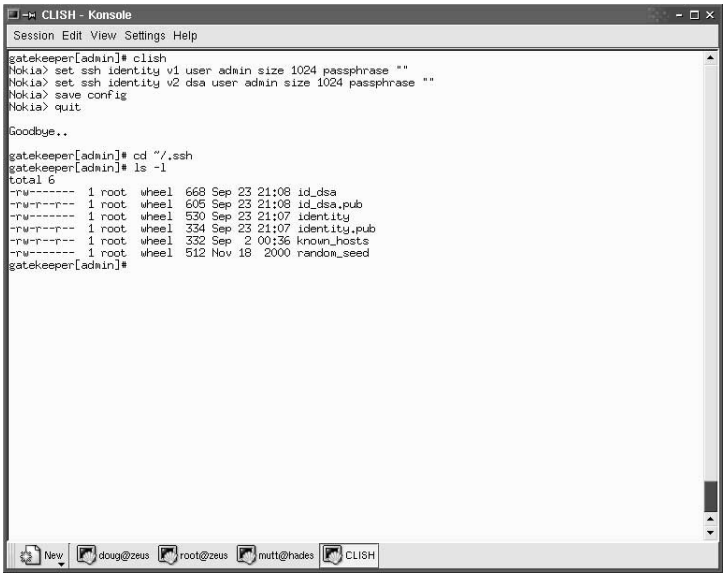
although the keys generated by *ssh-keygen* will not be visible in Voyager if you do so. CLISH is a good option for generating the public and private keys, because it is simple to use and any changes you make while in CLISH can be written to the global configuration file /config/active. We recommend generating version one and two keys because doing so will support the widest range of SSH servers.

The CLISH commands are as follows:

```
Nokia> set ssh identity v1 user admin size 1024 passphrase ""
Nokia> set ssh identity v2 dsa user admin size 1024 passphrase ""
Nokia> save config
Nokia> quit
```

Figure 8.6 shows a CLISH session that generates SSH version one and version two key pairs for the admin user, followed by a listing of the /var/admin/.ssh directory, where you can see the key pairs listed.

Figure 8.6 Generating SSH Key Pairs in CLISH



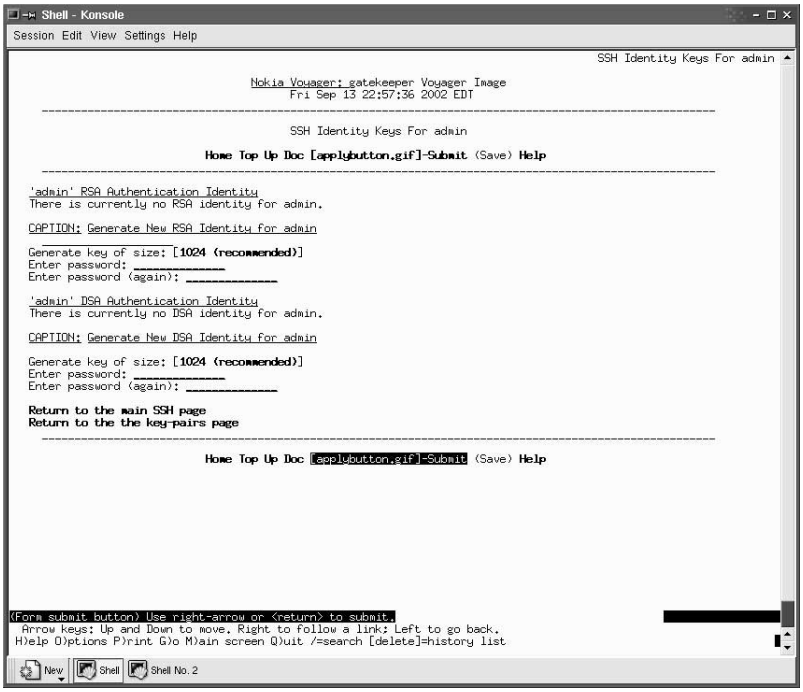
Here are descriptions of the files listed in Figure 8.6:

- **identity** RSA (SSH version one) private key file.
- **identity.pub** RSA (SSH version one) public key file.
- **id_dsa** DSA (SSH version two) private key file.
- **id_dsa.pub** DSA (SSH version two) public key file.

234_nokia_08.qxd 3/3/03 3:04 PM Page 260

If you must use lynx to generate the SSH keys because you are using an IPSO version prior to 3.6, open up a lynx session and select the **Config** link on the opening screen. From there, navigate to **SSH (Secure Shell) | Go to the key pairs page | View/Create Identity Keys for User 'admin'**. You should be at the screen titled SSH Identity Keys For admin, as shown in Figure 8.7.

Figure 8.7 Generating SSH Keys in lynx



Keep the default key size of 1024 bits, and leave the password fields empty. Apply your changes, and you should see the generated keys on the screen after a brief pause, as shown in Figure 8.8.

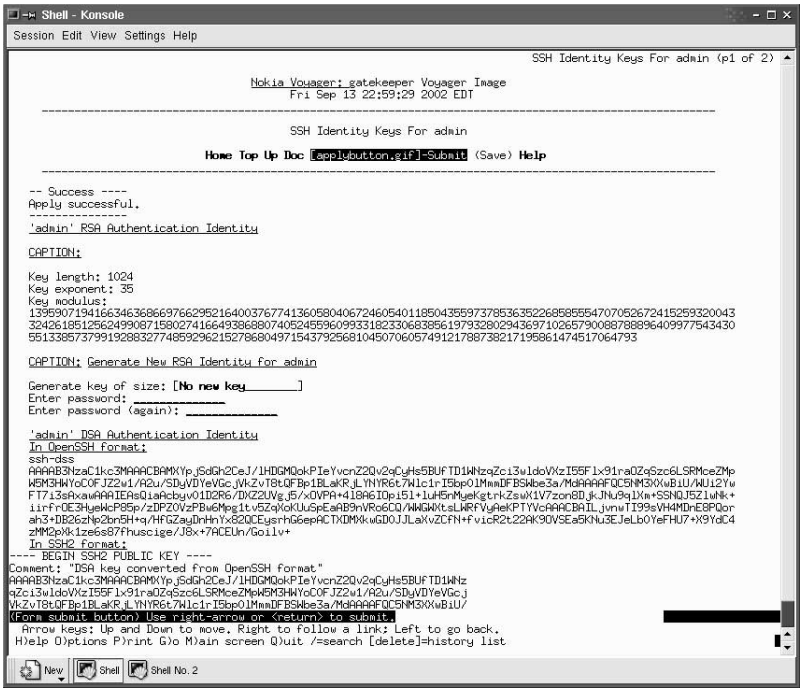
Save your changes before exiting lynx, and then type **cd ~/.ssh**. You should see the SSH key files listed after you type an **ls –l**. Note that the two public key files are just ASCII text, so you can use *cat* to view their contents (see Figure 8.9).

The next step is to copy the two public key files /var/admin/.ssh/identity .pub and /var/admin/.ssh/id_dsa.pub to the remote host that will be accepting your transfers. You can copy the files to a diskette or use SCP to transfer the files. It is even possible to completely avoid a file transfer and cut and paste the public key text if you have a session to each host open in a console window or xterm. Both keys should be appended to the end of .ssh/authorized_keys in the user

234_nokia_08.qxd 3/3/03 3:04 PM Page 261

account that will be accepting the log transfers. Be careful not to overwrite this file if it exists, because it could contain other public keys from other SSH clients and removing them will disable logins for those hosts. Once this is done, you should test connectivity from your Nokia device to the host with the SSH server. Assuming the remote host has an IP address of 10.100.6.153, and you copied the public key files to dmaxwell's user account, for example, you can type **ssh dmaxwell@10.100.6.153** to open a remote session. The first time you connect, you will be prompted to accept the host key for 10.100.6.153, which, when accepted, will be appended to your ~/.ssh/known_hosts file. You should connect and be dumped into a command shell as user dmaxwell without being prompted for a password, since the SSH client and server have authenticated each other with your public and private keys.

Figure 8.8 Generated SSH Keys

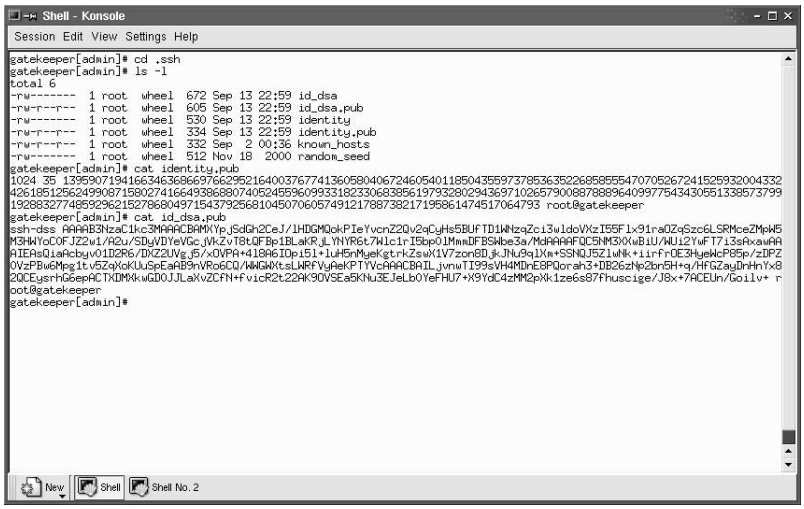


NOTE

You can read more about the basic theory behind public-key cryptography in Chapter 10 of *Check Point NG Next Generation Security Administration,* published by Syngress Publishing, ISBN 1928994741.

234_nokia_08.qxd 3/3/03 3:04 PM Page 262

Figure 8.9 SSH Keys in /var/admin/.ssh



Once you have cached the server's host key by logging in once, you can initiate unattended, encrypted file transfers with SCP. Assume you want to transfer all the firewall logs from your Nokia to the remote SSH server. You would use the following command to do so:

```
scp /var/fw/log/*log dmaxwell@10.100.6.153:
```

Note the trailing colon (:); this is necessary because SCP expects a target path specifier. In the command, the colon is shorthand for *Place the file in the home directory of the user dmaxwell*. All files matching the pattern **log* in your /var/fw/log directory will be transferred.

WARNING

Do not attempt the command while your management station is running, because the firewall processes have the log file opened for access continuously. Either run *fwstop* first, or perform an *fw logswitch* command, followed by an *fw logexport,* and compress your firewall logs before transferring them.

234_nokia_08.qxd 3/3/03 3:04 PM Page 263

Recovering Passwords

Any good firewall administrator's toolbox should have the answer to the question, "What do I do if I lose or forget the admin password?" Nokia has a procedure that allows you to reset the admin password if you have local console access. Here are the steps involved:

1. Boot into single-user mode from a directly attached console connection. In IPSO version 3.4.x and later, the console is considered "secure," so you will not be prompted for a password if you boot into single-user mode from the boot manager prompt. (See the section "Single-User Mode" earlier in this chapter.)

NOTE

For this procedure to work, you must boot into single-user mode from the boot manager prompt.

2. You will not be able to change the admin's password in lynx, because you need the old admin password to even start lynx. Nokia has an executable that will allow you to temporarily change the admin's password. Run **/etc/overpw** from the single-user shell. Follow the prompts (see the code that follows).
3. Type **reboot** or **Ctrl + D** and allow the system to come up into full multiuser mode. *If your version of IPSO is 3.5 or greater, skip to Step 7.*
4. In IPSO 3.4, a bug prevented the *overpw* command from working correctly, and it assigned an empty password to the admin user. You will not be prompted for an admin password in this case when the system comes up, enabling you to get to a command shell, but you still will not be able to log into Voyager because it does not accept empty passwords.
5. A fix for this situation is to run the command **dbpasswd admin *newpassword* "" ""**, where *newpassword* is your new password and the trailing """" specifies the old, empty password.
6. Run the command **dbset :save** to save the new password to the active configuration file.

234_nokia_08.qxd 3/3/03 3:04 PM Page 264

7. Once this procedure is complete, you should be able to use Voyager or lynx again to set a new, permanent admin password.

Here is what the above procedure looks like when run through on IPSO 3.6:

```
Enter pathname of shell or RETURN for sh:
# /etc/overpw
     This program is used to set a temporary admin password when you have
     lost the configured password.  You must have booted the machine into
     single user mode to run it.  The configured password will be changed.
     Please change the temporary password as soon as you log on to your
     system through voyager.

Please enter password for user admin:
Please re-enter password for confirmation:
Continue? [n] y
Running fsck...
/dev/rwd0f: clean, 65453 free (2461 frags, 7874 blocks, 0.6%
     fragmentation)
/dev/rwd0a: clean, 36154 free (42 frags, 4514 blocks, 0.1%
     fragmentation)
/dev/rwd0d: clean, 5802929 free (625 frags, 725288 blocks, 0.0%
     fragmentation)
/dev/rwd0e: clean, 902680 free (992 frags, 112711 blocks, 0.1%
     fragmentation)

     Admin password changed.  You may enter ^D to continue booting.
     THIS IS A TEMPORARY PASSWORD CHANGE.
     PLEASE USE VOYAGER TO CREATE A PERMANENT PASSWORD FOR THE USER ADMIN.
```

Using tcpdump

We saw above how *grep* and other command-line tools can be quite useful in foraging through firewall log files, but sometimes you need to see a more detailed view of network traffic. This is where *tcpdump* comes in. It is a network sniffer that can be used to display packet header information from various levels of IPSO's TCP stack.

The most basic use of *tcpdump* is to examine all the traffic across a network interface. It will show you a decent level of detail by default. The syntax for this

234_nokia_08.qxd 3/3/03 3:04 PM Page 265

task is *tcpdump –i <interface name>*. So, for example, if your Nokia's internal (physical) interface was named eth-s1p3, you would use **tcpdump –i eth-s1p3** to see all the traffic coming across that interface. This command usually gives you too much information, however, so *tcpdump* will accept Boolean expressions on its command line that will filter the output. You can specify hosts, ports, or protocols as part of the expression.

The easiest way to learn is to see some examples of *tcpdump* in use. Table 8.2 shows the most common *tcpdump* command line arguments, whereas Table 8.3 shows some useful command sequences you can try yourself. If you would like more information, the man page for *tcpdump* is one of the few available in IPSO. Don't forget that you can always use *grep* with the output of *tcpdump*, as we did with the log files earlier.

Table 8.2 tcpdump Command-Line Arguments

Option	Meaning
-a	Converts numeric addresses to names.
-c <number>	Stops after processing <*number*> packets.
-e	Displays the Layer 2 header information.
-i <interface name>	Listens on interface <*interface name*>.
-n	Don't convert numeric addresses to names.
-q	Prints less information.
-r <filename>	Reads packets from file <*filename*>, which was created with the *–w* option.
-S	Prints absolute, as opposed to relative, TCP sequence numbers.
-v, -vv, -vvv	Various levels of output verbosity, in increasing order.
-w <filename>	Writes output to the binary file <*filename*>.

Table 8.3 tcpdump Examples

This Command…	Will Show You…
tcpdump –i eth-s1p3 host 10.100.6.153 and not port 80	All the traffic to or from the host 10.100.6.153 that is not on port 80 over the interface eth-s1p3.

Continued

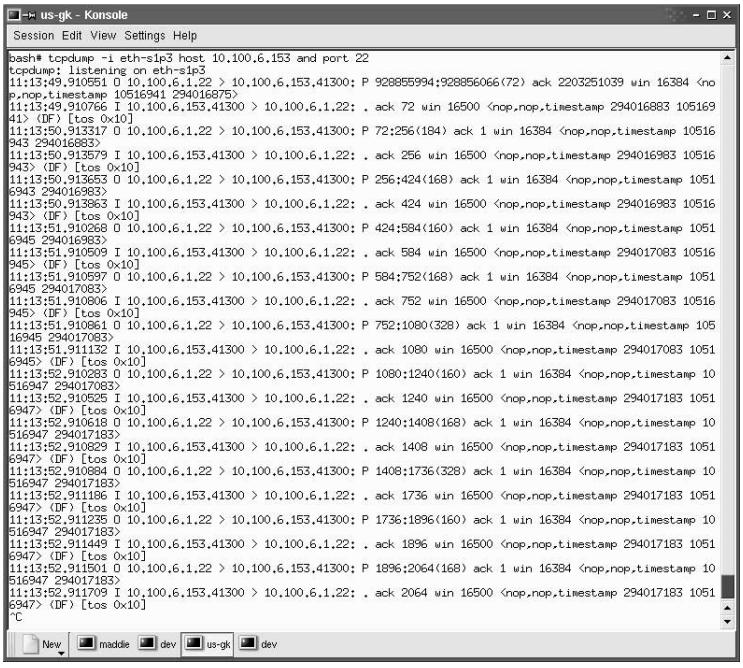
234_nokia_08.qxd 3/3/03 3:04 PM Page 266

Table 8.3 tcpdump Examples

This Command…	Will Show You…
tcpdump –i eth-s1p3 src host 10.100.6.153 and tcp dst port 22	All the TCP traffic to port 22 from the host 10.100.6.153 over the interface eth-s1p3.
tcpdump –i eth-s1p3 –w vrrp.out proto vrrp	All the VRRP traffic across the interface eth-s1p3. This will be written to the binary file *vrrp.out.*
tcpdump –i eth-s1p3 –r vrrp.out	All the data from the file *vrrp.out*, collected above.
tcpdump –i eth-s1p3 udp port 500	IKE key negotiation across the interface eth-s1p3.

In the commands described in Table 8.3, the output file generated by the *–w* switch is not a text file and can only be read by *tcpdump* or some other software written to parse the binary file format. If you want to save text output, redirect the standard output of these commands to a text file, as in *tcpdump –i eth-s1p3 host 10.100.6.153 and not port 80 > tcpdump.txt*. Figure 8.10 shows the output of an SSH session from my workstation at 10.100.6.153.

Figure 8.10 Output of tcpdump –i eth-s1p3 host 10.100.6.153 and port 22



234_nokia_08.qxd 3/3/03 3:04 PM Page 267

Let's analyze one of the lines in this output so you get an idea of what the information actually means. You should have a basic understanding of how TCP/IP works to get the most out of *tcpdump*.

```
11:13:49.910766 I 10.100.6.153.41300 > 10.100.6.1.22: . ack 72 win 16500
     <nop,nop,timestamp 294016883 10516941> (DF) [tos 0x10]
```

The first part, 11:13:49.910766, is a time stamp, which is as accurate as the IPSO kernel's internal clock. The part 10.100.6.153.41300 > 10.100.6.1.22 says that this packet came from 10.100.6.153, port 41300, and went to 10.100.6.1, port 22. The period (.) after the colon indicates that no TCP flags were set in this packet. We could see an S (SYN), F (FIN), P (PUSH), or R (RST) here as well. The string ack 72 means that this packet had a piggybacked ACK on it, for the 72nd data byte. win 16500 says that our available receive buffer is 16,500 bytes. The string <nop,nop,timestamp 294016883 10516941> describes various TCP options; the *nop* is a "no-op," used as padding. The only TCP option set in this packet is the timestamp option. The two numeric values are the timestamp value and timestamp echo reply. The (DF) means the "don't fragment" bit is on, and [tos 0x10] is the IP Type of Service option. In this case, 0x10 is the Minimize Delay option.

NOTE

An excellent reference on the inner workings of TCP/IP can be found in Richard Steven's book, *TCP/IP Illustrated*, Vol. I, ISBN 0201633469.

Troubleshooting Flows

Firewall *flows* are a feature Nokia added to the IPSO operating system to specifically work in conjunction with and increase the throughput for certain types of FireWall-1 traffic. To understand how flows work, we need to understand how IPSO and FireWall-1 normally work together to inspect and route packets at the kernel level.

The traditional name for the path that packets take through the IPSO kernel is *slowpath*. First, a packet comes in via a network interface and gets a kernel route lookup at the device driver level. It is then passed up to the FireWall-1 kernel module, where it is checked against the connection table. If the packet is not in the table, it gets checked against the firewall rule base. Assuming that the packet is part of an existing connection or is accepted by the rule base, it gets passed back

234_nokia_08.qxd 3/3/03 3:04 PM Page 268

down to the network interface device driver (now as an outbound packet), where it gets another route lookup. Then the connection table/route lookup process is repeated again, before the packet is sent on to its destination. This means that two connection table lookups and three routing table lookups for every packet. Nokia found that the bottleneck in this *slowpath* process was the connection table lookups, which are very slow compared with packet routing.

In an attempt to increase FireWall-1 throughput, Nokia implemented what is essentially a copy of the connection table at the device driver level. In what Nokia terms *flowpath*, incoming packets now get a connection table lookup at the same time they get a routing table lookup from the network interface device driver. The connection table lookup is done with the cached copy, not the FireWall-1 copy. If the packet matches an established connection, it is immediately forwarded on to its destination. If the packet does not match a connection table entry, it is passed up to the FireWall-1 kernel module, where it continues with the *slowpath* process as normal. Changes to the FireWall-1 connection table (for example, for new connections) are immediately propagated to the cached copy so that it is always up to date. This process results in a throughput increase for firewall traffic, with the best performance increase seen with small (64-byte) packets that are part of long-term, existing connections (such as large FTP or SCP transfers, for example). Nokia claims a four-fold increase in throughput in the best case, with performance gains decreasing as the packet size increases. A small amount of overhead is incurred when a new connection is established, because of the need to update the cached copy of the connection table, but this overhead is offset by the performance gains described, so throughput actually increases, even if only slightly, in almost all cases.

Firewall flows do have some limitations, however. Flows are not used at all for encrypted, ICMP, multicast, or authentication (security server) traffic. If you want to increase throughput of your VPN traffic, install a hardware accelerator. The combination of flows and a hardware VPN accelerator is the reason that Nokia's IP series devices attain such high firewall throughput numbers (refer back to Chapter 1). Note that NAT and antispoofing configuration data *is* cached by flows.

NOTE

Do not confuse Nokia's *flowpath* with Check Point's *fastpath*, which is a way to avoid connection table lookups in older FireWall-1 versions by assuming that all non-SYN packets were part of an existing connection and could be accepted without a connection table lookup. You could

234_nokia_08.qxd 3/3/03 3:04 PM Page 269

implement this at a global level by checking off the *fastpath* option in the version 4.0 policy editor. Although this feature resulted in some performance improvement, it was woefully insecure. Nokia's flows are a way to get an increase in throughput without any loss of security, since every packet gets the required connection table lookup.

There have also been problems with the implementation of flows in older versions of IPSO. The tool was introduced in IPSO 3.3 and has been enabled by default in all IPSO releases since then. Several Nokia knowledge base articles detail kernel panics that cause reboots when flows are enabled and certain conditions are met in FireWall-1 versions 4.1 SP2, SP3, and SP5. You should read Resolutions 4352 and 8731 in Nokia's knowledge base if you are using any of these versions. There are generally three solutions in each case:

1. Apply a hot fix or upgrade your particular FireWall-1 version, either of which can be downloaded from Check Point's support site. If you are on SP2 or SP3 now, SP4 will fix this problem. If you are on SP5 now, there is a hot fix available.
2. Upgrade to the latest FireWall-1 version and service pack available for your version of IPSO. At the time of this writing, this is SP6, with an OpenSSL hot fix. This is recommended because the latest service pack always includes past bug fixes and almost always fixes other bugs or security problems. Remember that not all versions of IPSO support all versions of FireWall-1, so you might need to upgrade your IPSO as well. (See Chapter 5 for details.)
3. Disable flows with the *ipsofwd slowpath* command. Nokia recommends this as a last resort.

You can always check whether flows are enabled with the *ipsofwd list* command:

```
usgk[admin]# ipsofwd list
net:ip:forward:noforwarding = 0
net:ip:forward:noforwarding_author =
net:ip:forward:switch_mode = flowpath
net:ip:forwarding = 1
usgk[admin]#
```

234_nokia_08.qxd 3/3/03 3:04 PM Page 270

You can see in this case that flows are enabled by looking at the net:ip: forward:switch_mode line, shown in bold type. If flows were disabled, the word *flowpath* would be replaced by *slowpath*.

Configuring & Implementing…

Understanding How IPSO Handles IP Forwarding

If Check Point's FireWall-1 is not installed on your Nokia appliance, IP forwarding is enabled by default, since most people would be using it as a router in that case. If FireWall-1 is installed, IP forwarding is disabled by default and, unlike other platforms supported by Check Point, the *cpconfig* utility *cannot* be used to control IP forwarding. The state of IP forwarding (either enabled or disabled) during the operation of the firewall is given by a few simple rules:

1. If your Check Point firewall is unable to load a policy from a management console when it starts and no previously loaded policy is stored on the firewall (which would be in $FWDIR/state), the system will load a default filter that blocks all inbound network connections. This includes the case where you unload your security policy with the *fw unload-local* (NG FP2 or later) or *fw unload* commands.
2. If FireWall-1 starts and loads a policy successfully, IP forwarding will be enabled.
3. When FireWall-1 is stopped with *fwstop* or *cpstop*, IP forwarding will be disabled.

To manually enable IP forwarding, perhaps for network testing after an *fw unload* command has been issued, use the following command: *ipsofwd on admin.*

To manually disable IP forwarding, use the command *ipsofwd off admin.*

The *admin* in both commands specifies the user who is changing IP forwarding. You can see the current forwarding value, the status of Nokia's flows, and the last person who changed the state of IP forwarding using the command *ipsofwd list*.

234_nokia_08.qxd 3/3/03 3:04 PM Page 271

Using ipsoinfo

The term *ipsoinfo* refers to the command that Nokia provides you to assist Nokia or your support provider in troubleshooting, should the need arise. It is similar in some respects to Check Point's *cpinfo* (or *fwinfo*), but *ipsoinfo* includes much more information. In fact, if the *cpinfo* command is found when the *ipsoinfo* command is run, it includes the *cpinfo* output in the file it generates. All you need to do to run the command is type **ipsoinfo** and press **Enter** when logged in as the admin user; after a while, the command will complete and leave a tarred, gzipped file named ipsoinfo-hostname-mm.dd.yyyy-time.tar.gz in the /var/admin directory. Be warned that this file can be quite large and can take some time to generate. If you extract it (either with WinZip after offloading it or with tar and gzip), it will contain the following files and directories:

- A system summary file
- A *dmesg* output
- A *cpinfo* output, if that command is present
- The ipsctl.log file, which tracks changes made through *ipsctl*
- The contents of the /var/etc, /var/log, /var/crash, and /var/tmp directories
- The contents of the /opt and /config/db directories

As you can see, this file will give you or your support provider lots of information about your system configuration—enough to recreate it exactly, if that is necessary. Figures 8.11 and 8.12 give you an idea of what the *ipsoinfo* process entails, with and without *cpinfo*, respectively. *cpinfo* does not come bundled with your Check Point software; you must download it from www.checkpoint.com/techsupport/downloading/utilities.html.

Using dmesg

The *dmesg* command outputs several screens of useful data, normally just the console messages printed by IPSO during its boot sequence. However, it also prints the most recent console errors on your running system. Figure 8.13 shows a partial output of the *dmesg* command on a smoothly running system—meaning that it has had no console errors that have displaced this information. In IPSO, the boot-time *dmesg* output is always saved in the file /var/run/dmesg.boot, so you can always look there if need be. You can see that your CPU type and speed, total RAM, and device probing results are all part of this output.

234_nokia_08.qxd 3/3/03 3:04 PM Page 272

Figure 8.11 ipsoinfo Run With cpinfo

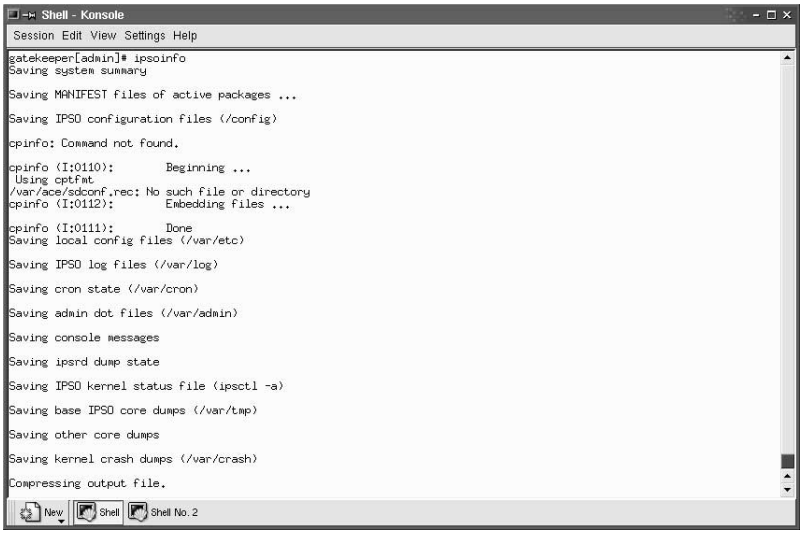


Figure 8.12 ipsoinfo Run Without cpinfo

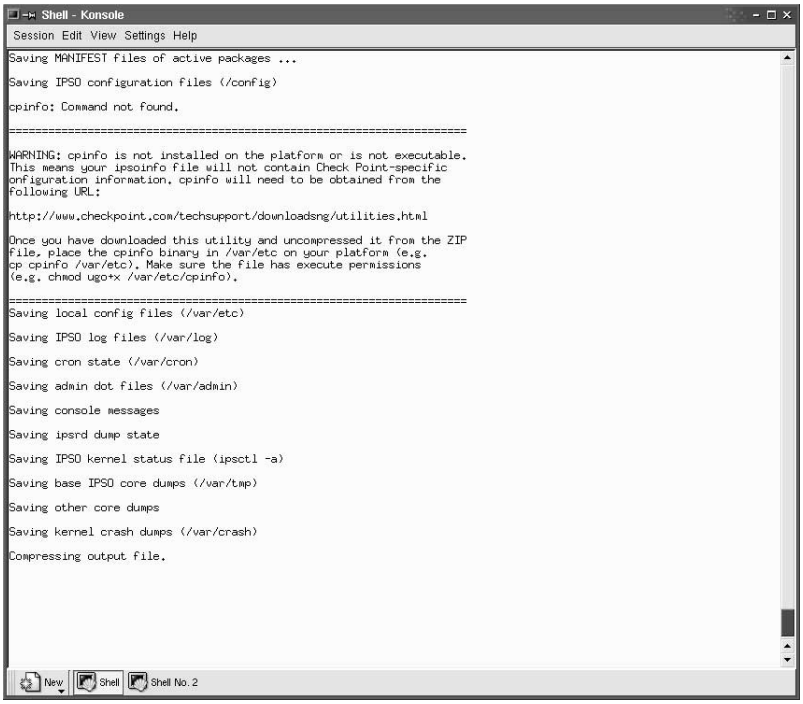


Figure 8.14 shows the partial *dmesg* output of a system experiencing many console errors, in this case on a Nokia system with Firewall-1 installed. This

234_nokia_08.qxd 3/3/03 3:04 PM Page 273

particular error message indicates a synchronization problem with Nokia flows and Check Point's state tables, for which a hot fix is available. These errors also appear in this system's log file, /var/log/messages, as critical errors.

Figure 8.13 Output of dmesg Command Showing Boot Sequence

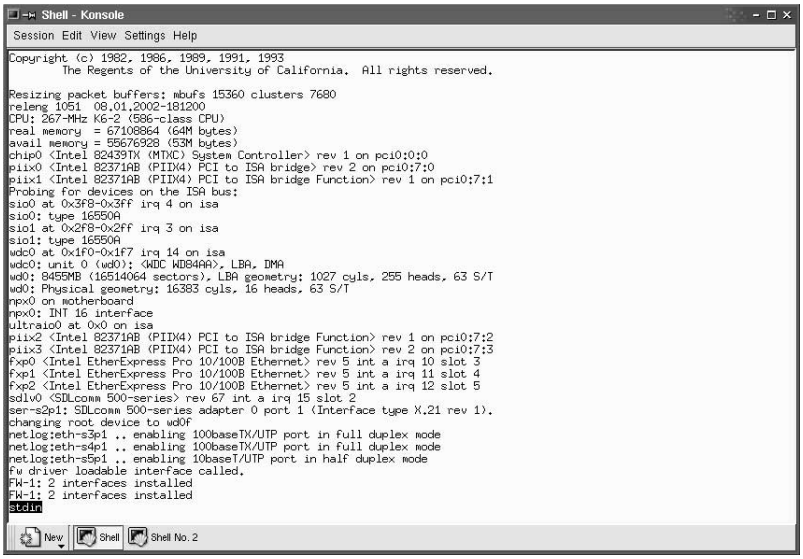
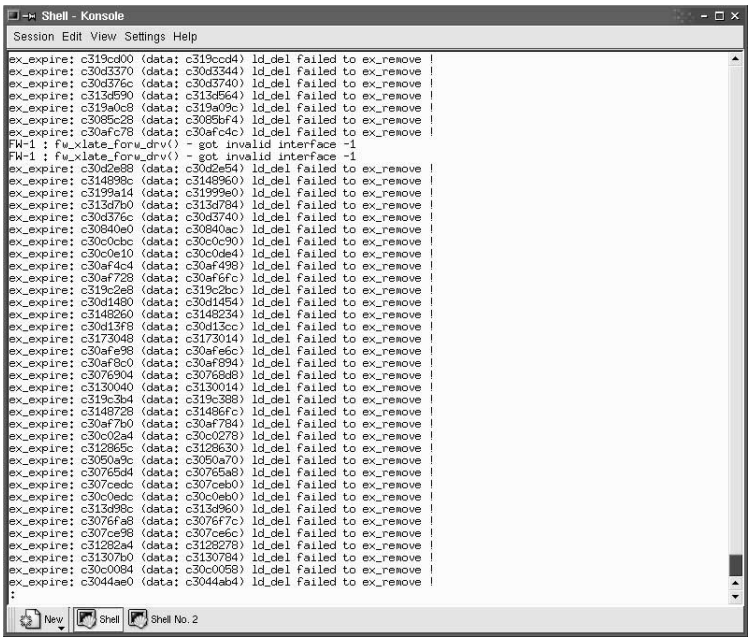


Figure 8.14 Output of dmesg Command Showing Console Errors



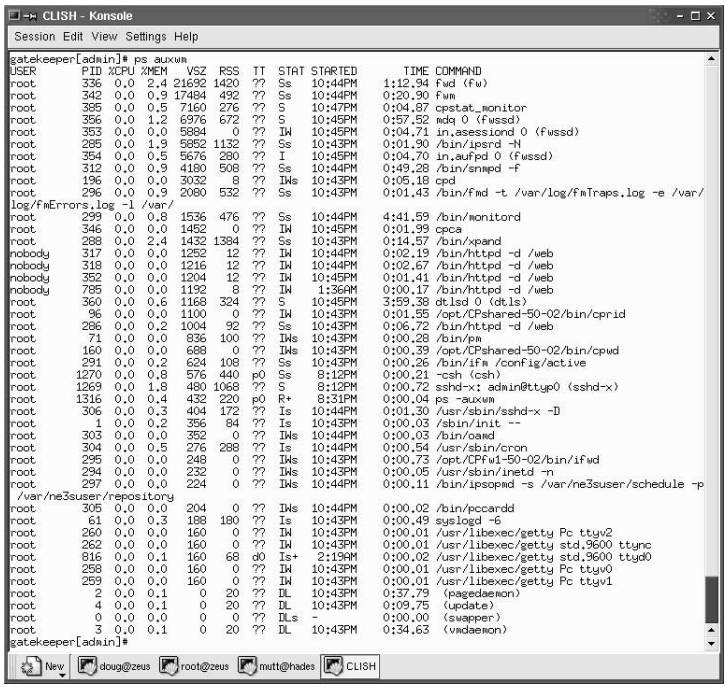
234_nokia_08.qxd 3/3/03 3:04 PM Page 274

Memory and Processes

In this section, we discuss memory and process monitoring from the command line. It is useful when you are troubleshooting any system to have a good grasp of what the system processes are actually doing behind the scenes, so we will tell you what some of IPSO's system daemons do. Where appropriate, we discuss CLISH and how to display this information from a CLISH shell.

The *ps* command gives you a useful snapshot of what is happening on your Nokia system, if it is used with the right arguments. Figure 8.15 shows the output of the *ps -auxwm* command, which displays statistics about all processes running on your Nokia device, in order of memory consumption.

Figure 8.15 Output of the ps -auxwm Command



What do these processes do? Some, like *csh,* have obvious functions, but others, like */bin/pm,* are more mysterious. The following is a list of the most important processes you are likely to see running on your Nokia appliance. They are presented in alphabetical order by process name:

- **cron** Executes scheduled commands as configured per user in a crontab file. The system crontab is in /etc/crontab.

234_nokia_08.qxd 3/3/03 3:04 PM Page 275

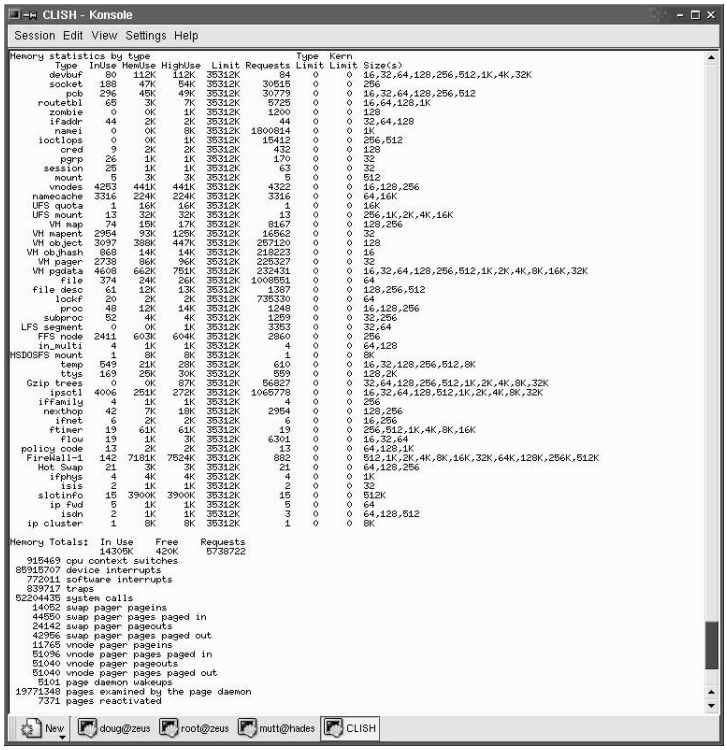
- **csh** The default command shell run when you log into IPSO remotely or via serial console. This can be changed; see Appendix B for a list of shells available on IPSO.
- **fmd** The Fault Management daemon.
- **getty** Monitors serial ports for user logins.
- **httpd** The Apache HTTP daemon as used by Voyager.
- **ifm** Monitors all interfaces for changes and maintains consistency with /config/active.
- **ifwd** Monitors network interfaces for changes and reloads the firewall security policy so that it enforces the policy on the new interfaces. Nokia recommends disabling this daemon on firewalls in its final interface configuration. This can be done through Voyager in the Check Point configuration section. *ifwd* can be useful on systems with hot-swappable NICs, such as the Nokia 600 or 700 series devices.
- **inetd** A daemon that listens for network connections for the servers it has enabled in its configuration file, /etc/inetd.conf. In a default Nokia installation, the only network server enabled is Telnet.
- **init** The first process started by the kernel after system boot. It starts all the *getty* processes that listen for console logins and also starts the *inetd* process that listens for network logins.
- **ipsrd** The IPSO routing daemon. It modifies routing tables after dynamic routing protocol updates or static route changes.
- **monitord** Collects real-time statistics on interfaces and routing.
- **pagedaemon** Manages the movement of memory pages into and out of main memory.
- **pccard** The PC Card Daemon. Handles the insertion and removal of PCMCIA cards.
- **pm** A daemon that monitors certain critical processes and restarts them if they die.
- **rcm** Controls the system's run level (single- or multiuser).
- **snmpd** A daemon that responds to queries via SNMP.
- **sshd-x** Secure Shell daemon. Listens for incoming secure shell connections.

234_nokia_08.qxd 3/3/03 3:04 PM Page 276

- **swapper** Manages the system swap space.
- **syslogd** Daemon that logs system messages.
- **update** This daemon flushes file system caches to disk periodically.
- **mdaemon** Memory daemon. It manages virtual memory.
- **xpand** Implements operating system changes to the global configuration file /config/active.

Sometimes during testing or network troubleshooting, it is useful to have a historical view of process and memory data. If you simply need a synopsis of memory and swap space usage at any given moment, you can use the command *vmstat –ism | more* to get very detailed memory and swap statistics, including how much memory each process in your system is using (see Figure 8.16). This form of *vmstat* is actually run as part of the *ipsoinfo* command.

Figure 8.16 Partial vmstat –ism Output



Look at the *vmstat* output. If the MemUse or HighUse column is greater than the Limit column for any of the categories, you might need to upgrade your physical memory.

234_nokia_08.qxd 3/3/03 3:04 PM Page 277

If you want to know which processes are routinely hogging your memory, CPU, or swap space, you can use the following script, written by Nokia and available in Nokia's knowledge base as Resolution 11093. It is designed to be run periodically from *cron* and determine which processes are using the most memory and swap space. When run, it appends its output to the file /var/admin/vmstatmon.out. Here is the *crontab* entry you should use:

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /var/admin/scripts/vmstatmon.sh
```

You can insert this line into the admin user's *crontab* using the *crontab –e* command, which opens a *Vi* editing session for you, or using Voyager or lynx if you are using IPSO 3.6. It is possible to use CLISH or Voyager to schedule cron jobs, but we recommend against using either one of them, since the interface is rather clumsy and doesn't appear to support schedules of finer than once per day without adding a separate job for each time increment.

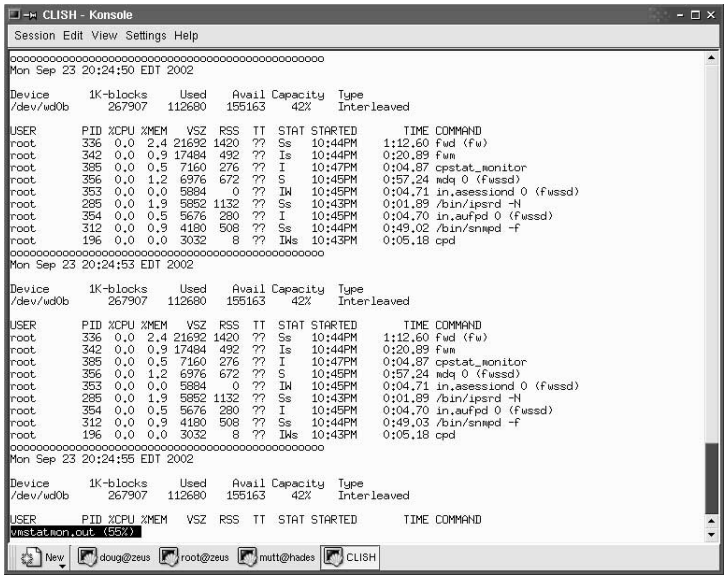
Make sure to save the following script as /var/admin/scripts/vmstatmon.sh, and make it executable with *chmod +x /var/admin/scripts/vmstatmon.sh:*

```
#!/bin/csh
source /var/etc/pm_cshrc
# This program will monitor the usage of the swap file over time
# and list the top 10 processes hogging memory
# Save the output to this file
set output=/var/admin/vmstatmon.out
# Number of processes to report on
set processnum=10
################## main program begin
# print the date
echo oooooooooooooooooooooooooooooooooooooooooooooooooo >>! $output
date >>! $output
echo " " >>! $output
# print the swap file usage
pstat -ks >>! $output
echo " " >>! $output
# print the top process hogs
ps auxmw | head -${processnum} >>! $output
```

Figure 8.17 shows the output of this script.

234_nokia_08.qxd 3/3/03 3:04 PM Page 278

Figure 8.17 Output of vmstatmon.sh



Designing & Planning…

Understanding IPSO Memory Allocation

The IPSO kernel allocates itself a fixed amount of RAM at system startup. This memory is reserved solely for the kernel's use and is *hardwired*, which means that it can never be swapped out to disk, unlike memory pages allocated to user processes. Just how much memory gets allocated depends on how much total RAM your system has. Approximately one-third of the total physical memory is reserved for the kernel's use, down to a minimum of 32MB. Applications (referring to processes in kernel stack and kernel module(s), if any) are each allowed to use up to 60 percent of the memory allocated to the kernel.

For example, on an IP330 with 64MB of RAM, the kernel will reserve the minimum of 32MB for itself, since one-third of 64 is approximately 21. Of that 32MB of kernel memory, 60 percent, or about 19MB, is available to any one application. The situation improves somewhat on an IP440 with 256MB of RAM; now the kernel reserves 85MB for itself and allows any one application to use up to 51MB.

Continued

234_nokia_08.qxd 3/3/03 3:04 PM Page 279

Why is this important? Because the Check Point inspection engine is implemented as a kernel module and needs kernel memory for its many state tables. In Check Point FireWall-1 version 4.1, the firewall kernel module didn't need much memory (relative to the amount reserved by the kernel itself), so a system with 64MB of RAM would run FireWall-1 just fine. Check Point NG requires more kernel memory and will run only adequately on systems with at least 128MB of RAM. You might have had to increase the amount of memory allocated to the firewall kernel module at some point, if you had a firewall with a large number of concurrent connections passing through it. You can see just how much memory the Check Point firewall module has allocated and is using by executing the command *fw ctl pstat*. The relevant section of the output is the one titled "Hash kernel memory statistics".

Note that IPSO imposes a maximum of 512MB for the kernel address space, regardless of system RAM available. This means a maximum of 170MB is available for any single application process.

234_nokia_08.qxd 3/3/03 3:04 PM Page 280

Summary

The IPSO boot manager is a valuable tool, knowledge of which is a large part of troubleshooting problems from boot failures to lost administrative passwords. Nokia puts the boot manager in flash memory in all its currently produced devices, although it has been on the system hard drive or even on a diskette in the past. The boot manager's main function is to load the IPSO kernel into main memory, and it does this if left to function unattended. If the boot sequence is interrupted, however, the boot manager gives you access to a rudimentary command shell, from which you can set and clear environment variables that control its function. The boot manager itself can be upgraded or reinstalled if the need arises.

Also extremely useful and new in IPSO 3.6 is the Command Line Interface Shell, or CLISH, which gives you the functionality of Voyager from a command shell. CLISH can be used in shell mode to enter interactive commands, or it can be used in batch- or single-command mode. Changes to the global configuration file /config/active can be made by saving your changes from the CLISH shell, much like the Save button in Voyager. It is possible for administrators to get exclusive access to the CLISH shell if they maintain a Nokia appliance with other administrators.

Nokia appliances can provide high levels of firewall throughput when *flows* are enabled. The biggest performance gain is seen in long-lasting connections with small packet sizes, such as large FTP transfers. *Flows* are enabled by default in IPSO 3.3 and later and can be disabled if needed with the command *ipsofwd slowpath*.

Troubleshooting problems can be made easier with the right command-line tools; anything that can be seen or done from a GUI interface such as Voyager or the Check Point log viewer is also possible from the shell. Managing and searching log files, in particular, benefits from the flexibility that tools such as *grep* offer. If your disk runs out of space or you need to more closely examine a log file, SSH (SCP) can be used to initiate secure, unattended log file transfers to remote hosts.

Other tools IPSO offers to make your life as an administrator easier include *ipsoinfo*, *dmesg*, *ps*, *vmstat*, and *tcpdump*. The *ipsoinfo* command is used to collect detailed system data for troubleshooting by your support provider or Nokia, whereas *dmesg* will show you boot-time messages or frequent console errors. The commands *ps* and *vmstat* can be used to diagnose memory or swap space problems, and *tcpdump* provides for detailed network packet header analysis.

234_nokia_08.qxd 3/3/03 3:04 PM Page 281

Solutions Fast Track

Understanding the Boot Manager

- ☑ The boot manager is responsible for loading the operating system kernel into memory.
- ☑ Nokia's boot manager is kept in flash memory, but it has been on the hard drive and diskettes, the latter in the IP400 series only.
- ☑ Single-user mode can be accessed through the boot manager with the *boot –s* command.
- ☑ You can change the operation of the boot manager by changing the value of various environment variables with the *setenv* command.
- ☑ The *set-defaults* command sets all the environment variables back to their default values.
- ☑ You might have to upgrade your boot manager prior to upgrading your IPSO version.
- ☑ You can upgrade or reinstall your boot manager from single-user mode.
- ☑ You can perform a factory-default installation using the *install* command or a boot diskette on the IP400 series of devices.

Using CLISH

- ☑ CLISH has all the functionality of Nokia's Voyager, from a command-line interface.
- ☑ CLISH can be run in shell mode or batch mode or used to execute single commands.
- ☑ CLISH provides context-sensitive help with the ? key and command completion with the Tab key..
- ☑ Remember to save your configuration changes with the *–s* flag or the *save config* command.
- ☑ It is possible to block other administrators from making changes with CLISH with the *set config-lock on* command.

234_nokia_08.qxd 3/3/03 3:04 PM Page 282

Troubleshooting

- ☑ Most of IPSO's system log files are kept in /var/log, except for firewall logs, which are in /var/fw/log.
- ☑ IPSO rotates system logs monthly so that they do not grow uncontrollably in size.
- ☑ *grep*, *more*, *tail*, and *gzip* can be used together to provide powerful log searching capabilities.
- ☑ SCP can be used to automate the transfer of log files in a secure way.
- ☑ It is possible to recover a lost admin password with the */etc/overpw* command.
- ☑ *tcpdump* can be used to show detailed packet-level data moving across a network interface.
- ☑ Firewall flows are enabled in IPSO 3.3 and higher, providing for increased FireWall-1 throughput for certain kinds of traffic.
- ☑ *Ipsoinfo* is used to generate a detailed system summary, which can be useful to your support provider.
- ☑ *Dmesg* shows you recent console errors and boot-time messages.
- ☑ The *ps* and *vmstat* commands can be used to troubleshoot memory or swap space problems.

234_nokia_08.qxd 3/3/03 3:04 PM Page 283

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: My SCP transfers still don't work, even though I followed your instructions. What can I do?

A: Pass the *–v* option to SCP. Try forcing one particular version of SSH with *–oProtocol=1* or *–oProtocol=2*. If you use strict mode checking in your SSH server's configuration file (usually in /etc/ssh/sshd_config), try restricting access to your SSH server's ~/.ssh/authorized_keys file with *chmod 0600 ~/.ssh/authorized_keys*.

Q: Why does my Nokia reboot itself after a "Fatal Trap 12" error message?

A: This message indicates that the IPSO kernel can't continue operating because a serious error has occurred. A *watchdog* timer present in all Nokia appliances causes an automatic reboot of the device if the kernel is unresponsive for any length of time. These errors can be caused by hardware (memory or disk) failures, misbehaving application programs, or even IPSO kernel bugs. In any case, these errors are worth a call to Nokia or your support provider.

Q: Can I increase the amount of kernel memory IPSO allocates to the FireWall-1 connection tables?

A: Yes, although this is rarely necessary in IPSO versions after 3.4. You need to use the modzap utility, which can be downloaded from Nokia's support site. See Nokia knowledge base Resolutions 1261 and 1325 for details and download links.

Q: I have an old version of IPSO and I want to upgrade, but I lost my admin password. Why doesn't /etc/overpw work?

A: IPSO versions 3.1.3 or before had a nonfunctioning *overpw* command. You have to call Nokia support to get an internal-only resolution for this problem.

234_nokia_08.qxd 3/3/03 3:04 PM Page 284